LNCS 3907

Franz Rothlauf et al. (Eds.)

Applications of Evolutionary Computing

EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT
EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC
Budapest, Hungary, April 2006, Proceedings

Springer

Franz Rothlauf et al. (Eds.)

Applications of Evolutionary Computing

EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT
EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC
Budapest, Hungary, April 10-12, 2006
Proceedings

Springer

Volume Editors

see next page

The cover illustration is the work of Pierre Grenier

Library of Congress Control Number: 2006922618

CR Subject Classification (1998): F.1, D.1, B, C.2, J.3, I.4, J.5

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN	0302-9743
ISBN-10	3-540-33237-5 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-33237-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11732242 06/3142 5 4 3 2 1 0

Volume Editors

Franz Rothlauf
Dept. of Business Administration and Information Systems
University of Mannheim
Schloss, 68131 Mannheim, Germany
rothlauf@uni-mannheim.de

Jürgen Branke
Institute AIFB
University of Karlsruhe
76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

Stefano Cagnoni
Dept. of Computer Engineering
University of Parma
Parco Area delle Scienze 181/a
43100 Parma, Italy
cagnoni@ce.unipr.it

Ernesto Costa
Dept. de Engenharia Informática
University of Coimbra
Polo II - Pinhal de Marrocos
3030 - 290 Coimbra, Portugal
ernesto@dei.uc.pt

Carlos Cotta
Dep. Lenguajes y Ciencias de la Computacion
Universidad de Malaga
29071 Malaga, Spain
ccottap@lcc.uma.es

Rolf Drechsler
Institute of Computer Science
University of Bremen
28359 Bremen, Germany
drechsle@informatik.uni-bremen.de

Evelyne Lutton
INRIA Rocquencourt
B.P. 105, 78153 LE CHESNAY Cedex, France
Evelyne.Lutton@inria.fr

Penousal Machado
Dep. de Engenharia Informática
University of Coimbra
Polo II, 3030 Coimbra, Portugal
machado@dei.uc.pt

Jason H. Moore
Dept. of Genetics
Dartmouth Medical School
HB7937 One Medical Center Dr.
Lebanon, NH 03756, USA
jason.h.moore@dartmouth.edu

Juan Romero
Facultad de Informatica
University of A Coruña
A Coruña, CP 15071, Spain
jj@udc.es

George D. Smith
School of Computing Sciences
University of East Anglia
UEA Norwich
Norwich NR4 7TJ, UK
gds@cp.uea.ac.uk

Giovanni Squillero
Dip. di Automatica e Informatica
Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 Torino, Italy
giovanni.squillero@polito.it

Hideyuki Takagi
Kyushu University, Faculty of Design
4-9-1, Shiobaru, Minami-ku
Fukuoka, 815-8540 Japan
takagi@design.kyushu-u.ac.jp

Preface

Evolutionary computation (EC) techniques are efficient nature-inspired planning and optimization methods based on the principles of natural evolution and genetics. Due to their efficiency and the simple underlying principles, these methods can be used for a large number of problems in the context of problem solving, optimization, and machine learning. A large and continuously increasing number of researchers and practitioners make use of EC techniques in many application domains. This book presents a careful selection of relevant EC applications combined with thorough examinations of techniques for a successful application of EC. The presented papers illustrate the current state of the art in the application of EC and should help and inspire researchers and practitioners to develop efficient EC methods for design and problem solving.

All the papers in this book were presented during EvoWorkshops 2006, which consisted of a varying collection of workshops on application-oriented aspects of EC. Since 1998, the format of the EvoWorkshops has proved to be very successful and to represent significant advances in the application areas of EC. As a result, over the last few years, EvoWorkshops has become one of the major events to focus solely on applicational aspects of EC, constituting an important link between EC research and the application of EC in a variety of domains.

EvoWorkshops is co-located with EuroGP, the main European event dedicated to genetic programming, and EvoCOP, which has become the main European conference on evolutionary computation in combinatorial optimization. The proceedings for both of these events, EuroGP 2006 and EvoCOP 2006, are also available in the LNCS series (number 3905 and 3906).

EvoWorkshops 2006, of which this volume contains the proceedings, was held in Budapest, Hungary, on April 10–12, 2006, jointly with EuroGP 2006 and EvoCOP 2006. EvoWorkshops 2006 consisted of the following i ndividual workshops:

– *EvoBIO*, the Fourth European Workshop on Evolutionary Bioinformatics,
– *EvoCOMNET*, the Third European Workshop on Evolutionary Computation in Communications, Networks, and Connected Systems,
– *EvoHOT*, the Third European Workshop on Evolutionary Computation in Hardware Optimization,
– *EvoIASP*, the Eighth European Workshop on Evolutionary Computation in Image Analysis and Signal Processing,
– *EvoINTERACTION*, the First European Workshop on Interactive Evolution and Humanized Computational Intelligence,
– *EvoMUSART*, the Fourth European Workshop on Evolutionary Music and Art, and

– *EvoSTOC*, the Third European Workshop on Evolutionary Algorithms in Stochastic and Dynamic Environments.

EvoBIO is concerned with the exploitation of EC and related techniques in bioinformatics and computational biology. For analyzing and understanding biological data, EC plays an increasingly important role in the pharmaceutical industry, in biotechnology, and in associated industries, as well as in scientific discovery.

EvoCOMNET addresses the application of EC techniques to problems in communications, networks, and connected systems. New communication technologies, the creation of interconnected communication and information networks such as the Internet, new types of interpersonal and interorganizational communication, and the integration and interconnection of production centers and industries are the driving forces on the road towards a connected, networked society. EC techniques are important tools for facing these challenges.

EvoHOT highlights the latest developments in the field of EC applications to hardware and design optimization. This includes various aspects like the design of electrical and digital circuits or the solving of classical hardware optimization problems.

EvoIASP, which was the first international event solely dedicated to the applications of EC to image analysis and signal processing, addressed this year topics ranging from fingerprinting to classification problems and artificial ants.

EvoInteraction deals with various aspects of interactive evolution, and more broadly of computational intelligence in interaction with human intelligence, including methodology, theoretical issues, and new applications.Interaction with humans raises several problems, mainly linked to what has been called the user bottleneck, i.e. human fatigue.

EvoMUSART focuses on the use of EC techniques for the development of creative systems. There is a growing interest in the application of these techniques in fields such as art, music, architecture, and design. The goal of EvoMUSART is to bring together researchers that use EC in this context, providing an opportunity to promote, present and discuss the latest work in the area, fostering its further developments and collaboration among researchers.

EvoSTOC addresses the application of EC in stochastic environments. This includes optimization problems with noisy and approximated fitness functions that are changing over time, the treatment of noise, and the search for robust solutions. These topics recently gained increasing attention in the EC community and EvoSTOC was the first workshop that provided a platform to present and discuss the latest research in this field.

EvoWorkshops 2006 continued the tradition of providing researchers in these fields, as well as people from industry, students, and interested newcomers, with an opportunity to present new results, discuss current developments and applications, or just become acquainted with the world of EC, besides fostering closer future interaction between members of all scientific communities that may benefit from EC techniques.

This year, the EvoWorkshops had the highest number of submissions ever. The number of submissions increased from 123 in 2004 to 143 in 2005 to 149 in 2006. EvoWorkshops 2006 accepted full papers with twelve pages and short papers with a reduced number of five pages. The acceptance rate of 43.6% for EvoWorkshops 2006 is an indicator for the high quality of the papers presented at the workshops and included in these proceedings. The following table gives some details on the number of submissions, the number of accepted papers, and the acceptance ratios for EvoWorkshops 2005 and EvoWorkshops 2006 (accepted short papers are in brackets). Of further importance for the statistics is the acceptance rate of EvoWorkshops 2004 which was 44.7%.

year	2006			2005		
	submissions	accept	ratio	submissions	accept	ratio
EvoBIO	40	21	52.5%	32	13	40.6%
EvoCOMNET	16	5	31.2%	22	5	22.7%
EvoHOT	9	5	55.6%	11	7	63.6%
EvoIASP	35	12(7)	34.3%	37	17	45.9%
EvoInteraction	8	6	75%	-	-	-
EvoMUSART	29	10(4)	34.5%	29	10(6)	34.5%
EvoSTOC	12	6(2)	50.0%	12	4(4)	33.3%
Total	149	65(13)	43.6%	143	56(10)	39.1%

We would like to thank all the members of the program committees for their quick and thorough work. We thank the Artpool Art Research Center of Budapest, and especially György Galántai, for offering space and expertise without which the wonderful evolutionary art and music exhibition associated with the conference would not have been possible. Furthermore, we would like to acknowledge the support from Napier University, Edinburgh.

Finally, we would like to say a special thanks to everybody who was involved in the preparation of the event. Special thanks are due to Jennifer Willies, whose work is a great and invaluable help. Without her support, running such a type of conference with a large number of different organizers and different opinions would be impossible. Further thanks go to the local organizer, Aniko Ekart, and her group, who made it possible to run such a conference in such a nice place.

April 2006

Franz Rothlauf Jürgen Branke Stefano Cagnoni
Ernesto Costa Carlos Cotta Rolf Drechsler
Evelyne Lutton Penousal Machado Jason H. Moore
Juan Romero George D. Smith Giovanni Squillero
Hideyuki Takagi

Organization

EvoWorkshops 2006 was jointly organized with EuroGP 2006 and EvoCOP 2006.

Organizing Committee

EvoWorkshops chair:	Franz Rothlauf, University of Mannheim, Germany
Local chair:	Aniko Ekart, Hungarian Academy of Sciences, Hungary
Publicity chair:	Steven Gustafson, University of Nottingham, UK
EvoBIO co-chairs:	Carlos Cotta, Universidad de Málaga, Spain Jason H. Moore, Darthmouth Medical School, USA
EvoCOMNET co-chairs:	Franz Rothlauf, University of Mannheim, Germany George D. Smith, University of East Anglia, UK
EvoHOT co-chairs:	Giovanni Squillero, Politecnico di Torino, Italy Rolf Drechsler, University of Bremen, Germany
EvoIASP chair:	Stefano Cagnoni, University of Parma, Italy
EvoInteraction co-chairs:	Evelyne Lutton, INRIA, France Hideyuki Takagi, Kyushu University, Japan
EvoMUSART co-chairs:	Juan Romero, University of A Coruña, Spain, Penousal Machado, University of Coimbra, Portugal
EvoSTOC co-chairs:	Jürgen Branke, University of Karlsruhe, Germany Ernesto Costa, University of Coimbra, Portugal

Program Committees

EvoBIO Program Committee:

Jesús Aguilar, Pablo de Olavide University, Spain
Jacek Błażewicz, Poznan University of Technology, Poland
Vincenzo Cutello, University of Catania, Italy
Gary Fogel, Natural Selection Inc., USA
James Foster, University of Idaho, USA
Alex Freitas, University of Kent, UK
Raúl Giráldez, Pablo de Olavide University, Spain
Rosalba Giugno, University of Catania, Italy
Jin-Kao Hao, University of Angers, France
Natalio Krasnogor, University of Nottingham, UK

Bill Langdon, University of Essex, UK
Robert MacCallum, Imperial College London, UK
Elena Marchiori, Vrije Universiteit Amsterdam, The Netherlands
Andrew Martin, University College London, UK
Pablo Moscato, University of Newcastle, Australia
Vic J. Rayward-Smith, University of East Anglia, UK
John Rowe, University of Birmingham, UK
Jem Rowland, University of Wales, UK
El-Ghazali Talbi, INRIA Futurs, France
Antoine van Kampen, Academic Medical Center, The Netherlands
Gwen Volkert, Kent State University, UK
Ray Walshe, Dublin City University, Ireland
Eckart Zitzler, ETH Zurich, Switzerland
Igor Zwir, University of Granada, Spain

EvoCOMNET Program Committee:

Stuart Allen, Cardiff University, UK
Jin-Kao Hao, University of Angers, France
Bryant Julstrom, St. Cloud State University, USA
Paul Marrow, BT UK, UK
Geoff McKeown, UEA Norwich, UK
Günther R. Raidl, Vienna University of Technology, Austria
Vic Rayward-Smith, UEA Norwich, UK
Franz Rothlauf, University of Mannheim, Germany
Giovanni Squillero, Politecnico di Torino, Italy
George D. Smith, University of East Anglia, UK
Andrew Tuson, City University, London, UK

EvoHOT Program Committee:

Varun Aggarwal, Massachusetts Institute of Technology, USA
Bernd Becker, Albert-Ludwigs-University Freiburg, Germany
Rolf Drechsler, University of Bremen, Germany
Michelanglo Grosso, Politecnico di Torino, Italy
Andrew Kinane, Dublin City University, Ireland
Gabriella Kókai, University of Erlangen, Germany
Una-May O'Reilly, Massachusetts Institute of Technology, USA
Mihai Oltean, Babeş-Bolyai University, Romania
Gregor Papa, Jozef Stefan Institute, Slovenia
Ernesto Sanchez, Politecnico di Torino, Italy
Lukáš Sekanina, Brno University of Technology, Czech Republic
Massimiliano Schillaci, Politecnico di Torino, Italy
Giovanni Squillero, Politecnico di Torino, Italy
Luca Sterpone, Politecnico di Torino, Italy
Andreas Veneris, University of Toronto, Canada

EvoIASP Program Committee:

EvoInteraction Program Committee:

EvoMUSART Program Committee:

Alan Dorin, Monash University, Australia
Alice C. Eldridge, University of Sussex, UK
Amilcar Cardoso, University of Coimbra, Portugal
Alejandro Pazos, University of A Coruna, Spain
Anargyros Sarafopoulos, Bournemouth University, UK
Andrew Horner, University of Science & Technology, Hong Kong
Antonino Santos, University of A Coruna, Spain
Bill Manaris, College of Charleston, USA
Carlos Grilo, School of Technology and Management of Leiria, Portugal
Colin Johnson, University of Kent, UK
Eduardo R. Miranda, University of Plymouth, UK
Evelyne Lutton, INRIA, France
Francisco Camara Pereira, University of Coimbra, Portugal
Gary Greenfield, University of Richmond, USA
Gerhard Widmer, Johannes Kepler University Linz, Austria
James McDermott, University of Limerick, Ireland
Janis Jefferies, Goldsmiths College, University of London, UK
Jeffrey Ventrella, Independent Artist, USA
John Collomosse, University of Bath, UK
Jon McCormack, Monash University, Australia
Jorge Tavares, University of Coimbra, Portugal
Ken Musgrave, Pandromeda, Inc., US
Lee Spector, Hampshire College, USA
Luigi Pagliarini, Academy of Fine Arts of Rome, Italy
& University of Southern Denmark, Denmark
Martin Hemberg, Imperial College London, UK
Matthew Lewis, Ohio State University, USA
Mauro Annunziato, Plancton Art Studio, Italy
Michael Young, University of London, UK
Niall J.L. Griffith, University of Limerick, Ireland
Paul Brown, Centre for Computational Neuroscience and Robotics,
University of Sussex, UK
Paulo Urbano, Universidade de Lisboa, Portugal
Peter Bentley, University College London, UK
Peter Todd, Max Planck Institute for Human Development, Germany
Rafael Ramirez, Pompeu Fabra University, Spain
Rodney Waschka II, North Carolina State University, USA
Scott Draves, San Francisco, USA
Stefano Cagnoni, University of Parma, Italy
Stephen Todd, IBM, UK
Tatsuo Unemi, Soka University, Japan
Tim Blackwell, University of London, UK
William Latham, Art Games Ltd., UK

EvoSTOC Program Committee:

Sponsoring Institutions

- EvoNet, the Network of Excellence on Evolutionary Computing
- Artpool Art Research Center, Budapest, Hungary

Table of Contents

Functional Classification of G-Protein Coupled Receptors, Based on Their Specific Ligand Coupling Patterns

Burcu Bakir[1] and Osman Ugur Sezerman[2]

[1] School of Biology, Georgia Institute of Technology, Atlanta, USA
[2] Sabanci University, Istanbul, Turkey

Abstract. Functional identification of G-Protein Coupled Receptors (GPCRs) is one of the current focus areas of pharmaceutical research. Although thousands of GPCR sequences are known, many of them remain as orphan sequences (the activating ligand is unknown). Therefore, classification methods for automated characterization of orphan GPCRs are imperative. In this study, for predicting Level 2 subfamilies of Amine GPCRs, a novel method for obtaining fixed-length feature vectors, based on the existence of activating ligand specific patterns, has been developed and utilized for a Support Vector Machine (SVM)-based classification. Exploiting the fact that there is a non-promiscuous relationship between the specific binding of GPCRs into their ligands and their functional classification, our method classifies Level 2 subfamilies of Amine GPCRs with a high predictive accuracy of 97.02% in a ten-fold cross validation test. The presented machine learning approach, bridges the gulf between the excess amount of GPCR sequence data and their poor functional characterization.

1 Introduction

G-Protein Coupled Receptors (GPCRs) are vital protein bundles with their key role in cellular signaling and regulation of various basic physiological processes. With their versatile functions in a wide range of physiological cellular conditions, they constitute one of the vastest families of eukaryotic transmembrane proteins [29]. In addition to the biological importance of their functional roles, their interaction with more than 50% of prescription drugs have lead GPCRs to be an excellent potential therapeutic target class for drug design and current pharmaceutical research. Over the last 20 years, several hundred new drugs have been registered which are directed towards modulating more than 20 different GPCRs, and approximately 40% of the top 200 synthetic drugs act on GPCRs [6]. Therefore, many pharmaceutical companies are involved in carrying out research aimed towards understanding the structure and function of these GPCR proteins. Even though thousands of GPCR sequences are known as a result of ongoing genomics projects [10], the crystal structure has been solved only for one GPCR sequence using electron diffraction at medium resolution (2.8

A) to date [15] and for many of the GPCRs the activating ligand is unknown, which are called orphan GPCRs [25]. Hence, based on sequence information, a functional classification method of those orphan GPCRs and new upcoming GPCR sequences is of great practical use in facilitating the identification and characterization of novel GPCRs.

Albeit laboratory experiments are the most reliable methods, they are not cost and labour effective. To automate the process, computational methods such as decision trees, discriminant analysis, neural networks and support vector machines (SVMs), have been extensively used in the fields of classification of biological data [21]. Among these methods, SVMs give best prediction performance, when applied to many real-life classification problems, including biological issues [30]. One of the most critical issues in classification is the minimization of the probability of error on test data using the trained classifier, which is also known as structural risk minimization. It has been demonstrated that SVMs are able to minimize the structural risk through finding a unique hyper-plane with maximum margin to separate data from two classes [27]. Therefore, compared with the other classification methods, SVM classifiers supply the best generalization ability on unseen data [30].

In the current literature, to classify GPCRs in different levels of families, there exist different attempts, such as using primary database search tools, e.g., BLAST [1], FASTA [20]. However, these methods require the query protein to be significantly similar to the database sequences in order to work properly. In addition to these database search tools, the same problem is addressed by using secondary database methods (profiles and patterns for classification), e.g., Attwood et al. have worked in particular on GPCRs in the PRINTS database [2] (whose data appeared in INTERPRO database [17]). Hidden Markov Models [24], bagging classification trees [32] and SVMs [13], [31] are other methods that have been used to classify GPCRs in different levels of families. Karchin et al. conducted the most comprehensive controlled experiments for sequence based prediction of GPCRs in [13] and showed that SVMs gave the highest accuracy in recognizing GPCR families. Whereas, in SVMs, an initial step to transform each protein sequence into a fixed-length vector is required and the predictive accuracy of SVMs significantly depends on this particular fixed-length vector. In [13], it is also pointed out that the SVM performance could be further increased by using feature vectors that encode only the most relevant features, since SVMs do not identify the features most responsible for class discrimination. Therefore, for an accurate SVM classification, feature vectors should reflect the unique biological information contained in sequences, which is specific to the type of classification problem.

In this paper, we address Level 2 subfamily classification of Amine GPCRs problem by applying Support Vector Machine (SVM) technique, using a novel fixed-length feature vector, based on the existence of activating ligand specific patterns. We obtain discriminative feature vectors by utilizing biological knowledge of the Level 2 subfamilies' transmembrane topology and identifying specific patterns for each Level 2 subfamily. Since these specific patterns carry ligand binding information, the features obtained from these patterns are more relevant features than amino acid and dipeptide composition of GPCR sequences, which in turn improves the accuracy of GPCR Level 2 subfamily classification. Applying our method on Amine Level 1 subfamily of GPCRs [10], we have shown that the classification accuracy is increased compared to the previous studies at the same level of classification.

2 Background

G-Protein Coupled Receptor Database (GPCRDB) information system organizes the GPCRs into a hierarchy of classes, Level 1 subfamilies (sub-families), Level 2 subfamilies (sub-sub-families), and types, based on the pharmacological classification of receptors [10]. A simplified view of GPCR family tree is presented in Figure 1. Since the binding of GPCRs into their specified ligands is important for drug design purposes, GPCRDB defines the classifications chemically (according to which ligands the receptor binds, based on the experimental data), rather than by sequence homology [13]. For class discrimination, generalization of the features shared by a diverse group of examples is required. Whereas, for subfamily discrimination, only the examples, that differ slightly, should be grouped together. Therefore, for GPCR subfamily classification problem, which is also related to GPCR function prediction, the ligand type that GPCR binds is more crucial than it is for GPCR class discrimination.

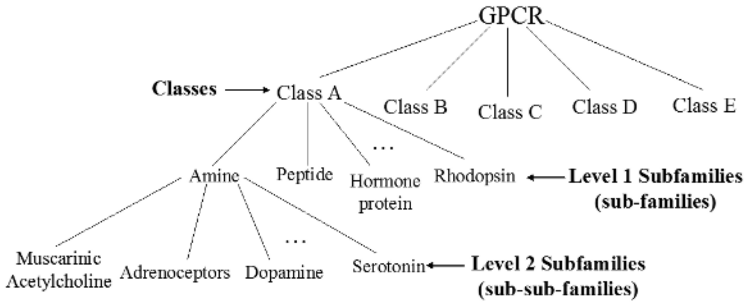


Fig. 1. Portion of GPCR family tree showing the five main classes of GPCRs and some subfamily members, based on the GPCRDB information system [10]

3 Materials and Methods

3.1 Benchmark Data

For GPCR function prediction from sequence information, subfamily recognition is more important than class recognition [13]. As mentioned before, subfamily recognition requires the knowledge of ligand coupling information of the receptor proteins. It is claimed that according to the binding of GPCRs with different ligand types, GPCRs are classified into at least six different families [9]. Among the sub-families in GPCRDB, Amine Level 1 subfamily of class A GPCRs is classified into seven sub-sub-families: (i) Muscarinic Acetylcholine, (ii) Adrenoceptors, (iii) Dopamine, (iv) Histamine, (v) Serotonin, (vi) Octopamine, (vii) Trace amine Level 2 subfamilies, according to the March 2005 release (9.0) of GPCRDB (Horn et al., 1998). Therefore, the correlation between sub-family classification and the specific binding of GPCRs to their ligands can be computationally explored for Level 2 subfamily classification of Amine Level 1 subfamily. Moreover, compared to the other classes, since Class A dominates by accounting for more than 80% of sequences as March 2005 release (9.0) of GPCRDB [10], it is the best studied class among different GPCRs. Thus, we will be able to compare our work with the previous studies. We use the same dataset, as that of Elrod and Chau, for Amine Level 1 subfamily GPCR sequences in GPCRDB, belonging to one of Acetylcholine, Adrenoceptor, Dopamine, Serotonin sub-sub-families, which have enough entries inside as a statistically significant training set, as shown in Table 1. The GPCR sequences in this dataset were extracted through the website http://www.expasy.org (SWISS-PROT database, Release 46.4, 2005) and fixed-length feature vectors are created for each sequence as it is explained in the next section.

Table 1. Summary of 168 Class A, Amine GPCRs, classified into four Level 2 Subfamilies as shown in [9]

Level 2 Subfamilies	Number of Sequences
Acetylcholine	31
Adrenoceptor	44
Dopamine	39
Serotonin	54
TOTAL	168

3.2 Fixed-Length Feature Vector Creation

Since protein sequences are of variable length, for classification, these sequences should be converted into fixed-length feature vectors. In order to obtain those fixed-length feature vectors, which also carry ligand specificity information, we followed a three step procedure as outlined in Figure 2. The first step, i.e., Topology Prediction step, aims to extract extracellular loop regions of the GPCR sequences since ligands couple to the outer loops of the GPCRs. So as to force fixed-length feature vectors to encode only biologically relevant features, activating ligand specificity information is taken into account. For this purpose, conserved patterns, which are specific to each sub-sub-family of GPCR sequences are found in extracellular GPCR sequences in step two, i.e., Pattern Discovery step. In third step, i.e., Pattern Matching step, the existence of those activating ligand specific (specific for a sub-sub-family) patterns is checked. So that we integrate the coupling specificity of GPCRs into their ligands knowledge into our

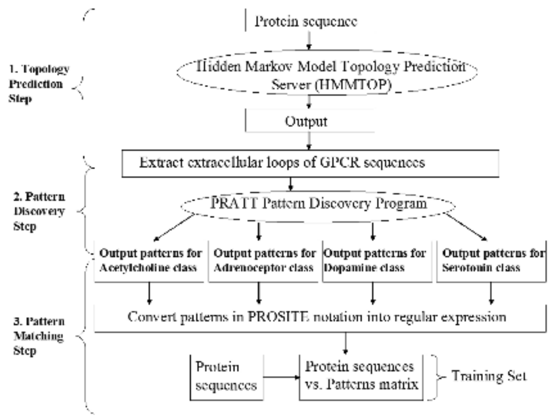

1. Topology Prediction Step
2. Pattern Discovery Step
3. Pattern Matching Step
Protein sequence
Hidden Markov Model Topology Prediction Server (HMMTOP)
Output
Extract extracellular loops of GPCR sequences
PRATT Pattern Discovery Program
Output patterns for Acetylcholine class
Output patterns for Adrenoceptor class
Output patterns for Dopamine class
Output patterns for Serotonin class
Convert patterns in PROSITE notation into regular expression
Protein sequences
Protein sequences vs. Patterns matrix
Training Set

Fig. 2. Flow chart for fixed-length feature vector creation

novel fixed-length feature vectors. Details of the three steps (Topology Prediction, Pattern Discovery, and Pattern Matching) for fixed-length feature vector creation are described below.

Topology Prediction. Since transmembrane (TM) topology pattern is shown to be well conserved among GPCRs that have the same function [19], for the 168 GPCR sequences in Elrod and Chau's dataset, TM topology is checked. For topology prediction, Hidden Markov Model for Topology Prediction (HMMTOP) server, which accurately predicts the topology of helical TM proteins, is used [26]. In order to segment amino acid sequences into membrane, inside, outside parts, HMMTOP method utilizes HMMs in a way that the product of the relative frequencies of the amino acids of these segments along the amino acid sequence is maximized. This shows that the maximum of the likelihood function on the space of all possible topologies of a given amino acid sequence, correlates with the experimentally established topology [25].

Following topology prediction, extracellular loop sequences are extracted for each 168 GPCR sequences, based on the fact that ligands couple to extracellular loops of GPCRs and we are interested in the relation between ligand specificity of GPCRs and GPCR sub-sub-family classification.

Pattern Discovery. In the second step of the fixed-length feature vector creation, for each sub-sub-family of GPCR sequences, flexible patterns that are conserved in the extracellular loop of that particular sub-sub-family GPCR sequences are found by using Pratt 2.1., flexible pattern discovery program [4]. Due to the flexibility of the Pratt patterns, they include ambiguous components, fixed and flexible wildcards, in addition to their identity components [12]. Hence, Pratt patterns are described using PROSITE notation [4].

Pratt finds patterns matching a subset of the input sequences. This subset is defined by "Min Percentage Seqs to Match (MPSM)" parameter, which defines the minimum percentage of the input sequences that should match a pattern. This threshold is set to 50% and 75% in this study in order not to allow for some very specific patterns that are not general to all GPCR sequences in any sub-sub-family. This can also be thought as a precaution to prevent overfitting problem. For each class of GPCRs, 50 conserved patterns are identified by two different MPSM parameters (50 and 75).

Pattern Matching. The final step for creating fixed-length feature vectors is to check for the existence of every activating ligand specific pattern in each outer GPCR sequence. In order to check the existence of the flexible Pratt patterns, all patterns in PROSITE notation are converted into regular expression form and then they are searched within 168 extracellular GPCR sequences. Consequently, by taking activating ligand specific pattern existence information into account, each GPCR sequence is represented with a vector in the 200 dimensional space (50 patterns multiplied by 4 output classes).

$$\mathbf{G_k} = (G_{k,1}, G_{k,2}, \ldots, G_{k,200}) \tag{1}$$

where $G_{k,1}$, $G_{k,2}$ $G_{k,200}$ are the 200 components of activating ligand specific pattern inclusion for the k^{th} extracellular GPCR sequence G_k. Note that if the k^{th} extracellular GPCR sequence has the pattern j, then $G_{k,j}$=1 and if the k^{th} extracellular GPCR sequence does not have the pattern j, then $G_{k,j}$=0, where j=1, 2, ... 200.

Writing down each fixed-length feature vector, G_k, in a new row, we obtain a $G_{k,j}$ matrix, where k=1, 2, ... 168; j=1, 2, ... 200. After insertion of the sub-sub-family labels for each of the GPCR sequences into the zeroth dimension of each G_k vector ($G_{k,0}$), the matrix corresponds to a training set. So that k=0, 1, 2, ... 168, where $G_{k,0}$ is 1, 2, 3 or 4, since four sub-sub-families are defined for this classification problem. Note that these 4 class output labelling (1, 2, 3, 4) does not imply any relationship between classes.

We have also created a second fixed-length feature vector, by using the best 10 patterns among the 50 patterns based on significance scores assigned by the Pratt program from each sub-sub-family. Using a similar representation, G_k is denoted in 40 dimensional space (10 patterns multiplied by 4 output classes), where j=1, 2, ... 40. A $G_{k,j}$ matrix is formed (similar to above), where k=1, 2, ... 168 and j=1, 2, ... 40 corresponding another training set.

As a result, four training sets (two training sets with 50 MPSM parameter, for j up to 40 or 200; another two with 75 MPSM parameter, for j up to 40 or 200) are created to produce a classifier using Support Vector Machines, as mentioned in detail below.

3.3 Classification Using Support Vector Machine (SVM)

The efficiency of SVMs for classification problems made them applicable in many real-world applications, including biological issues such as: protein classification and gene expression data analysis. SVM-based method for classification of sub-families of GPCRs, is first developed by Karchin et al. [13]. When applied to the problem of discriminating both Level 1 and Level 2 subfamilies of GPCRs, SVMs are shown to make significantly fewer errors of both false positive and false negative than WU-BLAST and SAM-T2K profile HMMs [13]. For these reasons, we selected to use SVMs for GPCRs' Level 2 subfamily classification problem.

4 Experiments

Since we are interested in the classification of Amine Level 1 sub-family into four Level 2 subfamilies, we are facing with a multi-class classification problem. We use LIBSVM software [7], which deals with multi-class classification problem implementing "one-against-one" approach. As suggested in [11], to be able to get satisfactory results, some preprocesses are performed before building a classifier using LIBSVM. Preprocesses, that are performed in this study, can be summarized in two headlines: i) Choice of Kernel function, ii) Grid search combined with cross-validation for parameter tuning.

4.1 Choice of Kernel Function

Among linear, polynomial, radial basis function (RBF) and sigmoid Kernel functions, RBF kernel is a reasonable first choice as stated in [11], [14], [16]. Therefore, grid search and parameter tuning is done on RBF kernels. However, results obtained by using those four kernels are compared with parameter tuned RBF kernel at the end.

4.2 Cross-Validation and Grid Search

In order to get better accuracy using RBF kernel for SVM classification, penalty parameter of error term, C, and γ parameter, which is specific to RBF kernel, should be tuned. Grid search procedure identifies the best (C, γ) pair, so that using these parameters the classifier (model) can accurately predict unseen test data [11]. Since the accuracy on test data also depends on the examples in the test data, cross validation is a better choice to tune (C, γ) parameters and select the best model that neither overfits nor underrepresents the training data. Compared to other advanced methods for parameter tuning, grid-search is straightforward, easy to implement and its computational time is not much more than advanced methods. Additionally, since each (C, γ) is independent, it can be easily parallelized. During grid search, it is recommended to try exponentially growing sequences of (C, γ) to identify good parameters [11].

To be able to solve our multi-class SVM classification problem, for each of our four training sets, grid search is performed for $C = 2^{-5}, 2^{-4}, \ldots, 2^{10}$ and $\gamma = 2^{5}, 2^{4}, \ldots, 2^{-10}$. Figure 3 shows the grid search with 10-fold cross validation for the training data with 200 attributes and 50 MPSM parameter. As it is seen in Figure 3, highest cross-validation accuracy is reached when γ=2^{-4} and C=(2^{0}, 2^{10}). After two preprocessing steps, as mentioned above, we build our classifiers using RBF kernel with best (C, γ) parameter pair, which is specific to the training set. Since we do not have a test set, and the number of examples in the training set is not big enough to separate into two, 10-fold cross-validation is done for each of the four training sets. Combining our biologically relevant fixed-length feature vector definition with a robust kernel, RBF, and parameter tuning with a grid search technique shows promising results, which is analyzed more in detail in the next section.

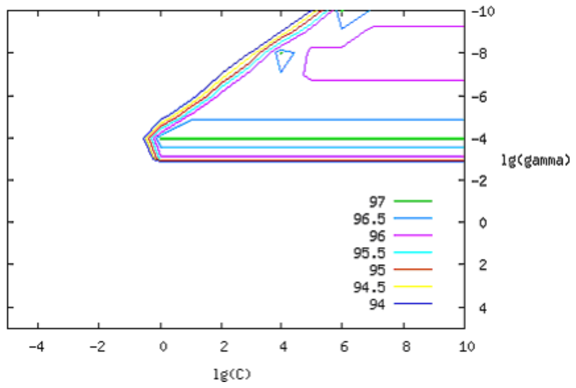


Fig. 3. Coarse grid search on C and with 10-fold cross validation for the training data with 200 attributes and 50 MPSM parameter. Highest cross-validation accuracy is obtained when $\gamma=2^{-4}$ and C=(2^0, 2^{10}).

5 Results

As mentioned before, in addition to the SVM classification with parameter tuned RBF kernel, other three standard kernel functions are tested as well (with their default parameters) on our four training data using 10-fold cross validation. Results for each experiment are summarized in Table 2.

Classification with RBF kernel with parameter tuning clearly outperforms other kernel functions in all cases. Since linear kernel is the specialized form of RBF kernel, results obtained with these two kernels without parameter tuning are quite close. Although, the classification accuracy with 200 and 40 attributes are so close, accuracy with 200 attributes are consistently better than with 40 attributes. The probable reason behind this observation is that 40 attributes are not enough to represent the examples (more attributes are needed to discriminate between data points), or those chosen 40 attributes do not correctly reflect the data points. In contrast to the strict domination of 200 attributes over 40 attributes, there is no such a relationship between training data with 50 MPSM parameter and 75 MPSM parameter. While sometimes one performs better, it is vice versa (e.g. results of RBF Kernel and RBF* Kernel in Table 2). Lower accuracy for the training data with 75 MPSM parameter is caused by overfitting, which decreases accuracy at the end whereas with 50 MPSM parameter, patterns that are conserved in at least 50% of the data can not represent overall data.

In this paper, for the classification of Level 2 Subfamily of Amine GPCR's, 97.02% prediction accuracy is achieved in a ten-fold cross validation. Compared to the existing GPCR functional classification methods, for the classification of Level 2 Subfamily of Amine GPCR's, our result is superior to the SVM method with a simpler fixed-length feature vector definition and no parameter tuning, where prediction accuracy is 94.01% [31] and covariant discriminant algorithm, where prediction accuracy is 83.23% [9]. In another study, Ying et al. performs classification for both sub-family and sub-sub-family levels of GPCRs using bagging classification tree [32] . For sub-sub-family level, using the same dataset [9], our prediction accuracy in a ten-fold cross validation (97.02%) is higher than their prediction accuracy obtained in ten-fold cross validation (82.4%). More extensive comparison with previous studies is presented in the following section.

Table 2. Results for four different training sets, as explained in the text, using four different kernel functions and RBF kernel with parameter tuning (RBF*), with 10-fold cross-validation

# of Attributes	MPSM Parameter	Linear Kernel	Polynomial Kernel	Sigmoid Kernel	RBF Kernel	RBF* Kernel
200	75	94.0476	48.8095	91.6667	91.6667	95.2381
200	50	96.4286	32.1429	86.3095	90.4762	97.0238
40	75	89.2857	47.0238	84.5238	86.3095	92.8571
40	50	92.8571	32.1479	84.5238	85.7143	93.4524

6 Discussion

The difference of this study from previous studies can be emphasized in two main points:

i) Fixed-length feature vector creation: We developed a novel method for obtaining fixed-length feature vectors of SVM. The naive idea that using direct protein sequence information as feature vector can not be used in SVM classification since the sequence length is not fixed. Many studies [9], [32], [31] attempted this problem by defining a fixed-length feature vector based on the protein's amino acid composition. Following the representation in [8], each protein is represented by a vector, X_k, in 20 dimensional space, where each dimension corresponds to how many times that particular amino acid, which represents that specific dimension, occurred in those particular protein.

$$\mathbf{X_k} = (X_{k,1}, X_{k,2}, \ldots, X_{k,20}) \tag{2}$$

where $X_{k,1}$, $X_{k,2}$ ... $X_{k,20}$ are 20 components of amino acid composition for the k^{th} protein X_k. In addition to the amino acid composition, in some of the studies, fixed-length vector is obtained by dipeptide composition [3], which takes local order of amino acids into account, in addition to the information about the fraction of amino acids. The dipeptide composition of each protein is shown using fractions of all possible dipeptides, where fraction of dipeptide i is the ratio of the number of dipeptide i in the protein divided by the total number of all possible dipeptides, namely 400. Alternatively, each protein sequence can also be transformed into a fixed-length feature vector, in the form of Fischer score vector [13].

Since in this study, the effect of activating ligand specificity in Level 2 Subfamily classification of Amine GPCRs is emphasized, a new feature vector is built, based on this observation. In this regard, we have used the existence information of activating ligand specific patterns, as fixed-length feature vectors, in order to come up with a biologically meaningful and distinctive measure. Therefore, the superiority of our feature vector stems from the biological importance of ligand coupling specificity for Level 2 Subfamily classification of Amine GPCRs. By combining those feature vectors with a robust kernel function, and parameter tuning strategy, we come up with an accurate classification method.

ii) Classification level: Apart from the definition of the feature vector for SVM, the exact classification level that we concentrate on, has been attempted in a few previous studies. Ying and Yanda and Ying et al. attempted the classification problem in the same Level 2 Subfamily of Amine GPCRs by using SVMs with amino acid composition as feature vector [31] and bagging classification tree [32], respectively. Our difference with their work is based on our novel feature vector definition as it is mentioned above, which in turn significantly affects the prediction accuracy (from 82.4% to 97.02% and 94.01% to 97.02% respectively). Apart from these particular papers, most of the previous studies concentrate on Superfamily level or Level 1 Subfamily. Although Karchin et al. have done experiments by using hierarchical multi-class SVMs, on Level 2 Subfamily [13] , they combine Class A Level 2 Subfamilies with Class C Level 2 Subfamilies.

Performance results in this study are promising and outperform other competitive techniques that classify GPCRs at the same level, with a very high cross validation accuracy of 97.02%. This result is mainly due to the definition of our feature vectors, since compared studies do not take into account such conserved pattern information for proper functioning of the GPCR. As the importance of specific ligand binding into GPCRs and the hidden information behind this binding is pointed out previously [13], we realized the use of ligand specific coupling patterns for creation of fixed-length feature vectors, which answers the need for biologically relevant features. Using these vectors for SVM classification and doing grid search for model selection, the accuracy have further improved even with very few sequences. With such accurate and automated GPCR functional classification methods, we are hoping to accelerate the pace of identifying proper GPCRs to facilitate drug discovery especially for schizophrenic and psychiatric diseases. Therefore, one of our future goals is to automate the presented procedure and come up with an integrated environment to perform GPCR classification conveniently with many flexible options to the biological users, who are not experts on the topic.

References

1. Altshul, S. et al.: Basic local alignment search tool. J. Mol. Biol. **215** (1990) 403–410
2. Attwood, T.K. et al.: PRINTS and its automatic supplement, prePRINTS. Nucleic Acids Research **31** (2003) 400–402
3. Bhasin, M. and Raghava, G.: GPCRpred: an SVM-based method for prediction of families and subfamilies of G-protein coupled receptors. Nucleic Acids Research **32** (2004) 383–389
4. Brazma, A. et al.: Discovering patterns and subfamilies in biosequences. Proceedings of the Fourth International Conference on Intellignent Systems for Molecular Biology (ISMB-96), AAAI Press (1996) 34–43
   Pratt 2.1 software is available at www.ebi.ac.uk/pratt
5. Byvatov, E. and Schneider, G.: Support vector machine applications in bioinformatics. Appl. Bioinformatics **2** (2003) 67–77
6. Chalmers, D.T. and Behan, D.P.: The Use of Constitutively Active GPCRs in Drug Discovery and Functional Genomics. Nature Reviews, Drug Discovery **1** (2002) 599-608
7. Chang, C.C. and Lin, C.J.: LIBSVM : a library for support vector machines. (2001) LIBSVM software is available at http://www.csie.ntu.edu.tw/ cjlin/libsvm
8. Chou, K.C.: A Novel Approach to Predicting Protein Structural Classes in a (ZO-l)-D Amino Acid Composition Space. PROTEINS: Structure, Function, and Genetics **21** (1995) 319–344
9. Elrod, D.W. and Chou, K.C.: A study on the correlation of G-protein-coupled receptor types with amino acid composition. Protein Eng. **15** (2002) 713–715
10. Horn, F. et al.: GPCRDB: an information system for G protein coupled receptors. Nucleic Acids Res. **26** (1998) 275–279
    Available at: www.gpcr.org/7tm
11. Hsu, C.W. et al.: A Practical Guide to Support Vector Classification. Image, Speech and Intelligent Systems (ISIS) Seminars. (2004)
12. Jonassen, I. et al.: Finding flexible patterns in unaligned protein sequences. Protein Sci. **4** (1995) 1587–1595
13. Karchin, R. et al.: Classifying G-protein coupled receptors with support vector machines. Bioinformatics **18** (2001) 147–159
14. Keerthi, S.S. and Lin, C.J.: Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation **15** (2003) 1667-1689
15. Krzysztof, P. et al.: Crystal Structure of Rhodopsin: A G- Protein-Coupled Receptor. Science **4** (2000) 739–745
16. Lin, H.T. and Lin, C.J.: A study on sigmoid kernels for SVM and the train ing of nonPSDkernels by SMO type methods. Technical report, Department of Computer Science and Information Engineering, National Taiwan University. (2003)
    Available at http://www.csie.ntu.edu.tw/ cjlin/papers/tanh.pdf
17. Mulder, N.J. et al.: The InterPro Database - 2003 brings increased coverage and new features. Nucleic Acids Research **31** (2003) 315–318
18. Neuwald, A. and Green, P.: Detecting Patterns in Protein Sequences. J. Mol. Biol. **239** (1994) 698-712
19. Otaki, J.M. and Firestein, S.: Length analyses of mammalian g-protein-coupled receptors. J. Theor. Biol. **211** (2001) 77–100
20. Pearson, W. and Lipman, D.: Improved tools for biological sequence analysis. Proceedings of National Academic Science **85** (1988) 2444-2448

21. Quinlan, J.R.: C4.5; Programs for Machine Learning, Morgan Kauffman Publishers, San Mateo, CA (1988)
22. Sadka, T. and Linial, M.: Families of membranous proteins can be characterized by the amino acid composition of their transmembrane domains. Bioinformatics **21** (2005) 378–386
23. Schoneberg, T. et al.: The structural basis of g-protein-coupled receptor function and dysfunction in human diseases. Rev Physiol Biochem Pharmacol. **144** (2002) 143–227
24. Sreekumar, K.R. et al.: Predicting GPCR-G-Protein coupling using hidden Markov models. Bioinformatics **20** (2004) 3490–3499
    Swiss-Prot database (Release 46.4, 2005) is available at http:// www.expasy.org
25. Tusndy, G.E. and Simon, I.: Principles Governing Amino Acid Composition of Integral Membrane Proteins: Applications to topology prediction. J. Mol. Biol. **283** (1998) 489–506
26. Tusndy, G.E. and Simon, I.: The HMMTOP transmembrane topology prediction server. Bioinformatics **17** (2001) 849–850
    Available at: http://www.enzim.hu/hmmtop
27. Vapnik, V. The Nature of Statistical Learning Theory, Springer-Verlag, New York. (1995)
28. Vert,J.P. Introduction to Support Vector Machines and applications to computational biology.(2001)
29. Vilo, J. et al.: Prediction of the Coupling Specificity of G Protein Coupled Receptors to their G Proteins. Bioinformatics **17** (2001) 174–181
30. Yang, Z.R.: Biological applications of support vector machines. Brief. Bioinform. **5** (2004) 328–338
31. Ying, H. and Yanda, L.: Classifying G-protein Coupled Receptors with Support Vector Machine. Advances in Neural Network (ISNN 2004), Springer LNCS. **3174** (2004) 448–452
32. Ying, H. et al.: Classifying G-protein Coupled receptors with bagging classification tree. Computational Biology and Chemistry **28** (2004) 275–280

Incorporating Biological Domain Knowledge into Cluster Validity Assessment

Nadia Bolshakova[1], Francisco Azuaje[2], and Pádraig Cunningham[1]

[1] Department of Computer Science, Trinity College Dublin, Ireland
{nadia.bolshakova, padraig.cunningham}@cs.tcd.ie
[2] School of Computing and Mathematics, University of Ulster, Jordanstown, BT37 0QB, UK
fj.azuaje@ulster.ac.uk

Abstract. This paper presents an approach for assessing cluster validity based on similarity knowledge extracted from the Gene Ontology (GO) and databases annotated to the GO. A knowledge-driven cluster validity assessment system for microarray data was implemented. Different methods were applied to measure similarity between yeast genes products based on the GO. This research proposes two methods for calculating cluster validity indices using GO-driven similarity. The first approach processes overall similarity values, which are calculated by taking into account the combined annotations originating from the three GO hierarchies. The second approach is based on the calculation of GO hierarchy-independent similarity values, which originate from each of these hierarchies. A traditional node-counting method and an information content technique have been implemented to measure knowledge-based similarity between genes products (biological distances). The results contribute to the evaluation of clustering outcomes and the identification of optimal cluster partitions, which may represent an effective tool to support biomedical knowledge discovery in gene expression data analysis.

1 Introduction

Over the past few years DNA microarrays have become a key tool in functional genomics. They allow monitoring the expression of thousands of genes in parallel over many experimental conditions (e.g. tissue types, growth environments). This technology enables researchers to collect significant amounts of data, which need to be analysed to discover functional relationships between genes or samples. The results from a single experiment are generally presented in the form of a data matrix in which rows represent genes and columns represent conditions. Each entry in the data matrix is a measure of the expression level of a particular gene under a specific condition.

A central step in the analysis of DNA microarray data is the identification of groups of genes and/or conditions that exhibit similar expression patterns. Clustering is a fundamental approach to classifying expression patterns for biological and biomedical applications. The main assumption is that genes that are contained in a particular functional pathway should be co-regulated and therefore should exhibit similar patterns of expression [1]. A great variety of clustering algorithms have been developed for gene expression data. The next data analysis step is to integrate these numerical analyses of co-expressed genes with biological function information. Many approaches and tools have been proposed to address this problem at different processing levels. Some methods, for example, score whole clustering outcomes or specific clusters according to their biological relevance, other techniques aim to estimate the significance of over-represented functional annotations, such as those encoded in the Gene Ontology (GO), in clusters [2], [3], [4], [5]. Some approaches directly incorporate biological knowledge (e.g. functional, curated annotations) into the clustering process to aid in the detection of relevant clusters of co-expressed genes involved in common processes [6], [7]. Several tools have been developed for ontological analysis of gene expression data (see review by Khatri and Drăghici [8], for instance) and more tools are likely to be proposed in the future.

The prediction of the correct number of clusters in a data set is a fundamental problem in unsupervised learning. Various cluster validity indices have been proposed to measure the quality of clustering results [9], [10]. Recent studies confirm that there is no universal pattern recognition and clustering model to predict molecular profiles across different datasets. Thus, it is useful not to rely on one single clustering or validation method, but to apply a variety of approaches. Therefore, combination of GO-based (knowledge-driven) validation and microarray data (data-driven) validation methods may be used for the estimation of the number of clusters. This estimation approach may represent a useful tool to support biological and biomedical knowledge discovery.

We implemented a knowledge-driven cluster validity assessment system for microarray data clustering. Unlike traditional methods that only use (gene expression) data-derived indices, our method consists of validity indices that incorporate similarity knowledge originating from the GO and a GO-driven annotation database. We used annotations from the *Saccharomyces Genome Database* (SGD) (October 2005 release of the GO database). A traditional node-counting method proposed by Wu and Palmer [11] and an information content technique proposed by Resnik [12] were implemented to measure similarity between genes products. These similarity measurements have not been implemented for clustering evaluation by other research.

The main objective of this research is to assess the application of knowledge-driven cluster validity methods to estimate the number of clusters in a known data set derived from *Saccharomyces cerevisiae.*

2 The GO and Cluster Validity Assessment

The automated integration of background knowledge is fundamental to support the generation and validation of hypotheses about the function of gene products. The GO and GO-based annotation databases represent recent examples of such knowledge resources. The GO is a structured, shared vocabulary that allows the annotation of gene products across different model organisms. The

GO comprises three independent hierarchies: molecular function (MF), biological process (BP) and cellular component (CC). Researchers can represent relationships between gene products and annotation terms encoded in these hierarchies. Previous research has applied GO information to detect over-represented functional annotations in clusters of genes obtained from expression analyses [13]. It has also been suggested to assess gene sequence similarity and expression correlation [14]. For a deeper review of the GO and its applications, the reader is referred to its website (http://www.geneontology.org) and Wang et al. [14].

Topological and statistical information extracted from the GO and databases annotated to the GO may be used to measure similarity between gene products. Different GO-driven similarity assessment methods may be then implemented to perform clustering or to quantify the quality of the resulting clusters. Cluster validity assessment may consist of data- and knowledge-driven methods, which aim to estimate the optimal cluster partition from a collection of candidate partitions [15]. Data-driven methods mainly include statistical tests or validity indices applied to the data clustered. A data-driven, cluster validity assessment platform was previously reported by Bolshakova and Azuaje, [9], [10]. We have previously proposed knowledge-driven methods to enhance the predictive reliability and potential biological relevance of the results [15].

Traditional GO-based cluster description methods have consisted of statistical analyses of the enrichment of GO terms in a cluster. Currently, there is a relatively large number of tools implementing such an approach [8]. At the same time, this approach is severely limited in certain regards (for detailed review on ontological analysis see by Khatri and Drăghici [8]). For instance, overestimation of probability values describing over-representation of terms. This may be due to the lack of more complete knowledge or the incorporation of biased datasets to make statistical adjustments and detect spurious associations. However, the application of GO-based similarity to perform clustering and validate clustering outcomes has not been widely investigated. A recent contribution by Speer et al. [16], [17] presented an algorithm that incorporates GO annotations to cluster genes. They applied data-driven Davies-Bouldin and Silhouette indices to estimate the quality of the clusters.

This research applies two approaches to calculating cluster validity indices. The first approach processes overall similarity values, which are calculated by taking into account the combined annotations originating from the three GO hierarchies. The second approach is based on the calculation of independent similarity values, which originate from each of these hierarchies. The second approach allows one to estimate the effect of each of the GO hierarchies on the validation process.

3 GO-Based Similarity Measurement Techniques

For a given pair of gene products, g_1 and g_2, sets of GO terms $T_1 = t_i$ and $T_2 = t_j$ are used to annotate these genes. Before estimating between-gene similarity it is first necessary to understand how to measure between-term similarity. We implemented GO-based between-term similarity using a traditional approach proposed by Wu and Palmer [11] and an information content technique proposed by Resnik [12].

3.1 Wu and Palmer-Based Method

Similarity was defined by Wu and Palmer [11] as follows:

$$sim(t_i, t_j) = \frac{2N}{N_i + N_j + 2N} \tag{1}$$

where N_i and N_j are the number of links (edges) from t_i and t_j to their closest common parent in the GO hierarchy, T_{ij}, and N is the number of links from T_{ij} to the GO hierarchy root.

This similarity assessment metric may be transformed into a distance, d, metric:

$$d(t_i, t_j) = 1 - sim(t_i, t_j) \tag{2}$$

then the average inter-set similarity value across each pair of t_i and t_j is computed [13]:

$$d(g_k, g_m) = \underset{i,j}{\text{avg}}(d(t_{ki}, t_{mj})) \tag{3}$$

This between-term distance aggregation may then be used as an estimate of the GO-based similarity between two genes products g_k and g_m, which is defined as:

$$d(g_k, g_m) = \underset{i,j}{\text{avg}}(1 - \frac{2N}{N_{ki} + N_{mj} + 2N}) \tag{4}$$

3.2 Resnik-Based Similarity Measurement

This similarity was defined by Resnik [12] as follows:

$$sim(t_i, t_j) = max(-log(p(T_{ij}))) \tag{5}$$

T_{ij} is defined as above and has the highest information value V defined as $-log(p(T_{ij}))$,where $p(T_{ij})$ is a probability, of finding term T_{ij} (or its descendants) in the dataset of genes under study, i.e. the SGD in this study.

Such similarity assessment metric may be transformed into a distance metric:

$$d(t_i, t_j) = \frac{1}{1 + sim(t_i, t_j)} \tag{6}$$

Based on the average value across each pair of t_i and t_j, as computed by Azuaje and Bodenreider [13], the GO-based similarity between two genes products g_1 and g_2 is defined as:

$$d(g_k, g_m) = \underset{i,j}{\text{avg}}(\frac{1}{1 + max(-log(p(T_{kmij})))}) \tag{7}$$

In this research we first study an approach based on the aggregation of similarity information originating from all three GO. We also proposed and implemented three hierarchy - specific similarity assessment techniques, each based on information individually extracted from each GO hierarchy (BP, MF and CC).

4 Clustering and Cluster Validation Methods

4.1 Clustering

The data analysed in this paper comprised yeast genes described by their expression values during the cell cycle [18]. Previous research has shown that disjoint clusters of genes are significantly associated with each of the five cell cycle stages: **early G1, late G1, S, G2, M**. Several cluster partitions (with numbers of clusters from two to six clusters), obtained with the k-means algorithm, were analysed to estimate the optimum number of clusters for this dataset. Clustering was performed with the Machaon CVE tool [10].

4.2 Cluster Validation Methods

Cluster validation was performed using two validity indices: the C-index [19] and the Goodman-Kruskal index [20], whose data-driven versions have been shown to be effective cluster validity estimators for different types of clustering applications. Nevertheless, each of the implemented validation methods has their advantages and limitations. For example, Goodman-Kruskal index is expected to be robust against outliers because quadruples of patterns are used for its computation. However, its drawback is its high computational complexity in comparison, for example, with the C-index.

C-index. The *C-index* [19], C, is defined as follows:

$$C = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{8}$$

where S, S_{min}, S_{max} are calculated as follows. Let p be the number of all pairs of samples (conditions) from the same cluster. Then S is the sum of distances between samples in those p pairs. Let P be a number of all possible pairs of samples in the dataset. Ordering those P pairs by distances we can select p pairs with the smallest and p pairs with the largest distances between samples. The sum of the p smallest distances is equal to S_{min}, whilst the sum of the p largest is equal to S_{max}. From this formula it follows that the nominator will be small if pairs of samples with small distances are in the same cluster. Thus, small values of C correspond to good clusters. We calculated distances using the knowledge-driven methods described above. The number of clusters that minimize *C-index* is taken as the optimal number of clusters, c.

Goodman-Kruskal index. For a given dataset, $X_j (j = 1, , k$, where k is the total number of samples (gene products in this application), j, in the dataset, this method assigns all possible quadruples [20]. Let d be the distance between any two samples (w and x, or y and z) in X_j. A *quadruple* is called *concordant* if one of the following two conditions is true:

$d(w, x) < d(y, z)$, w and x are in the same cluster and y and z are in different clusters.

$d(w, x) > d(y, z)$, w and x are in different clusters and y and z are in the same cluster.

By contrast, a *quadruple* is called *disconcordant* if one of following two conditions is true:

$d(w, x) < d(y, z)$, w and x are in different clusters and y and z are in the same cluster.

$d(w, x) > d(y, z)$, w and x are in the same cluster and y and z are in different clusters.

We adapted this method by calculating distances using the knowledge-driven methods described above.

A good partition is one with many *concordant* and few *disconcordant quadruples*. Let N_{con} and N_{dis} denote the number of *concordant* and *disconcordant quadruples*, respectively. Then the *Goodman-Kruskal index, GK*, is defined as:

$$GK = \frac{N_{con} - N_{dis}}{N_{con} + N_{dis}} \tag{9}$$

Large values of *GK* are associated with good partitions. Thus, the number of clusters that maximize the *GK index* is taken as the optimal number of clusters, c.

5 Results

The clustering algorithm was applied to produce different partitions consisting of 2 to 6 clusters each. Then, the validity indices were computed for each of these partitioning results. The two GO-based similarity assessment techniques introduced above were used for all cases to calculate biological distances between the genes.

Tables 1 to 4 show the predictions made by the validity indices at each number of clusters. Bold entries represent the optimal number of clusters, c, predicted by each method. In the tables the first cluster validity index approach processes overall GO-based similarity values, which are calculated by taking into account the combined annotations originating from the three GO hierarchies. The other indices are based on the calculation of independent similarity values, independently obtained from each of the GO hierarchies.

The C-indices based on Resnik similarity measurement and similarity information from the MF, BP and the combined hierarchies indicated that the optimal

Table 1. C-index predictions based on Wu and Palmer's GO-based similarity metric for expression clusters originating from yeast data

Validity indices based on:	c=2	c=3	c=4	c=5	c=6
Combined hierarchies	0.51	0.472	0.464	**0.453**	0.463
Biological process	0.501	0.321	0.259	**0.235**	0.237
Molecular function	0.501	0.32	0.274	**0.243**	0.272
Cellular component	**0.514**	0.586	0.602	0.614	0.615

Table 2. C-index values predictions based on Resnik's GO-based similarity estimation technique for expression clusters originating from yeast data

Validity indices based on:	c=2	c=3	c=4	c=5	c=6
Combined hierarchies	0.504	0.395	0.373	**0.349**	0.369
Biological process	0.503	0.321	0.261	**0.234**	0.243
Molecular function	0.501	0.32	0.278	**0.25**	0.29
Cellular component	**0.517**	0.645	0.69	0.723	0.759

Table 3. Goodman-Kruskal index values used Wu and Palmer's similarity metric for expression clusters originating from yeast data

Validity indices based on:	c=2	c=3	c=4	c=5	c=6
Combined hierarchies	-0.023	-0.01	-0.018	**0.004**	-0.017
Biological process	-0.013	0.005	-0.005	**0.034**	0.018
Molecular function	-0.02	0.009	0.005	**0.066**	-0.026
Cellular component	-0.025	**-0.022**	-0.032	-0.046	-0.025

Table 4. Goodman-Kruskal index values used Resnik's similarity metric for expression clusters originating from yeast data

Validity indices based on:	c=2	c=3	c=4	c=5	c=6
Combined hierarchies	-0.026	-0.001	-0.02	**0.016**	-0.01
Biological process	-0.018	0.014	-0.012	**0.055**	0.044
Molecular function	-0.02	0.012	0.004	**0.087**	-0.016
Cellular component	-0.025	-0.035	**-0.024**	-0.037	-0.025

number of clusters is $c = 5$, which is consistent with the cluster structure expected [18]. The C-indices based on Wu and Palmer similarity measurement and similarity information from the MF and BP indicated that the optimal number of clusters is $c = 5$. In all cases only the method based on the CC hierarchy suggested the partition with two clusters as the optimal partition, which confirms that cellular localization information does not adequately reflect relevant functional relationships in this dataset.

For the Goodman-Kruskal method again only the method based on the CC hierarchy suggested the partition different from $c = 5$ as the optimal partition.

6 Accompanying Tool

The approaches described in this paper are available as part of the *Machaon CVE* (Clustering and Validation Environment) [10]. This software platform has been designed to support clustering-based analyses of expression patterns including several data- and knowledge-driven cluster validity indices. The program and additional information may be found at http://www.cs.tcd.ie/ Nadia.Bolshakova/GOtool.html

7 Conclusion

This paper presented an approach to assessing cluster validity based on similarity knowledge extracted from the GO and GO-driven functional databases. A knowledge-driven cluster validity assessment system for microarray data clustering was implemented. Edge-counting and information content approaches were implemented to measure similarity between genes products based on the GO. Edge-counting approach calculates the distance between the nodes associated with these terms in a hierarchy. The shorter the distance, the higher the similarity. The limitation is that it heavily relies on the idea that nodes and links in the GO are uniformly distributed.

The research applies two methods for calculating cluster validity indices. The first approach process overall similarity values, which are calculated by taking into account the combined annotations originating from the three GO hierarchies. The second approach is based on the calculation of independent similarity values, which originate from each of these hierarchies. The advantage of our method compared to other computer-based validity assessment approaches lies in the application of prior biological knowledge to estimate functional distances between genes and the quality of the resulting clusters. This study contributes to the development of techniques for facilitating the statistical and biological validity assessment of data mining results in functional genomics.

It was shown that the applied GO-based cluster validity indices could be used to support the discovery of clusters of genes sharing similar functions. Such clusters may indicate regulatory pathways, which could be significantly relevant to specific phenotypes or physiological conditions.

Previous research has successfully applied C-index using knowledge-driven methods (GO-based Resnik similarity measure) [15] to estimate the quality of the clusters.

Future research will include the comparison and combination of different data- and knowledge-driven cluster validity indices. Further analyses will comprise, for instance, the implementation of permutation tests as well as comprehensive cluster descriptions using significantly over-represented GO terms.

The results contribute to the evaluation of clustering outcomes and the identification of optimal cluster partitions, which may represent an effective tool to support biomedical knowledge discovery in gene expression data analysis.

Acknowledgements

This research is partly based upon works supported by the Science Foundation Ireland under Grant No. S.F.I.-02IN.1I111.

References

1. Fitch, J., Sokhansanj, B.: Genomic engineering: Moving beyond DNA. Sequence to function. Proceedings of the IEEE **88** (2000) 1949–1971
2. Gat-Viks, I., Sharan, R., Shamir, R.: Scoring clustering solutions by their biological relevance. Bioinformatics **19** (2003) 2381–2389
3. Lee, S., Hur, J., Kim, Y.: A graph-theoretic modeling on go space for biological interpretation on gene clusters. Bioinformatics **20** (2004) 381–388
4. Goeman, J., van de Geer, S., de Kort, F., van Houwelingen, H.: A global test for groups of genes: testing association with a clinical outcome. Bioinformatics **20** (2004) 93–99
5. Raychaudhuri, S., Altman, R.: A literature-based method for assessing the functional coherence of a gene group. Bioinformatics **19** (2003) 396–401
6. Hanisch, D., Zien, A., Zimmer, R., Lengauer, T.: Co-clustering of biological networks and gene expression data. Bioinformatics **18** (2002) S145–S154
7. Sohler, F., Hanisch, D., Zimmer, R.: New methods for joint analysis of biological networks and expression data. Bioinformatics **20** (2004) 1517–1521
8. Khatri, P., Drăghici, S.: Ontological analysis of gene expression data: current tools, limitations, and open problems. Bioinformatics **21** (2005) 3587–3595
9. Bolshakova, N., Azuaje, F.: Cluster validation techniques for genome expression data. Signal Processing **83** (2003) 825–833
10. Bolshakova, N., Azuaje, F.: Machaon CVE: cluster validation for gene expression data. Bioinformatics **19** (2003) 2494–2495
11. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: 32nd Annual Meeting of the Association for Computational Linguistics, New Mexico State University, Las Cruces, New Mexico (1994) 133–138
12. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI). (1995) 448–453
13. Azuaje, F., Bodenreider, O.: Incorporating ontology-driven similarity knowledge into functional genomics: an exploratory study. In: Proceedings of the fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE 2004). (2004) 317–324
14. Wang, H., Azuaje, F., Bodenreider, O., Dopazo, J.: Gene expression correlation and gene ontology-based similarity: An assessment of quantitative relationships. In: Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, La Jolla-California, IEEE Press (2004) 25–31
15. Bolshakova, N., Azuaje, F., Cunningham, P.: A knowledge-driven approach to cluster validity assessment. Bioinformatics **21** (2005) 2546–2547
16. Speer, N., Spieth, C., Zell, A.: A memetic clustering algorithm for the functional partition of genes based on the Gene Ontology. In: Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2004), IEEE Press (2004) 252–259
17. Speer, N., Spieth, C., Zell, A.: Functional grouping of genes using spectral clustering and gene ontology. In: Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2005), IEEE Press (2005) 298–303
18. Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., Davis, R.: A genomewide transcriptional analysis of the mitotic cell cycle. Molecular Cell **2** (1998) 65–73

19. Hubert, L., Schultz, J.: Quadratic assignment as a general data-analysis strategy. British Journal of Mathematical and Statistical Psychologie (1976) 190–241
20. Goodman, L., Kruskal, W.: Measures of associations for cross-validations. Journal of Ameracan Statistical Association (1954) 732–764

A Novel Mathematical Model for the Optimization of DNA–Chip Design and Its Implementation

Kornélia Danyi[1], Gabriella Kókai[2], and József Csontos[3]

[1] Institute of Informatics, University of Szeged,
Árpad tér 2, H–6720 Szeged, Hungary
[2] Department of Programming Systems, Friedrich–Alexander University,
Martensstraße 3, D–91058 Erlangen, Germany
[3] Department of Biomedical Sciences, Creighton University,
2500 California Plaza Omaha, NE 68178, USA

Abstract. A variety of recent achievements in the field of biology, chemistry and information technology have made possible the development of DNA chips. They allow us to analyze the sequences and functions of different genes simultaneously and detect small differences in those. They are source of tremendous amount of data in the field of Bioinformatics. Moreover, the engineering process of DNA chip requires the latest results of information technology, too. In this paper, we address the mathematical problem of the prediction the hybridization process on the chip surface. A novel in situ in silico approach is presented and the obtained results are discussed.

1 Introduction

The rapid development of nanotechnology and computer science led to the emergence of a new research field, called *bioinformatics*. One of the most important technique of this new discipline is *DNA–chip* or *DNA–microarray*. It also represents a revolutionary innovation in the area of applied medical diagnostics. With the help of it the presence of pathogens and a predominant proportion of genetically based diseases can be detected parallel very quickly and accurately. In other words, it can extremely accelerate the precise diagnostics, so the appropriate treatment can be started earlier.

Nevertheless, the more extensive use of the method is nowadays limited by its high operational costs. For example the production of 80 homogeneous chips with 20,000 DNA fragments costs approximately € 40,000. The largest part of these expenses results from the production of the so–called *masks*, which are used to determine the chemical structure of DNA pieces. Obviously, the large manufacturing costs mean substantial disadvantage. Therefore, the designing process needs special attention. The aim of our work is to facilitate the engineering process and help to avoid extra charges which are due to chemicals carrying non–appropriate genetical information.

We proceed as follows: In section 2 a short introduction to DNA–chip technology and Simulated Annealing method is given. The estimation of the hybridization is summarized in section 3. Our mathematical model is presented together with its optimization in section 4. The test results can be found in section 5, and conclusion with future work plans in section 6.

2 DNA–Chips

DNA–chip itself is a solid carrier (glass or special plastic plate) with a set of single stranded *DNA fragments*, so–called *probes* on it. The *sample* is usually a solution which also contains *DNA fragments*. According to Watson and Crick a DNA molecule consists of two helically twisted strands connected together with a series of hydrogen bonds (*double–helix*) and each strand has 4 distinct building blocks (*nucleotides* or *bases*), adenine (*dA*), guanine (*dG*), cytosine (*dC*) and thymine (*dT*). Generally, the different basis sequences determine different genes, which are large DNA fragments, carry and transmit genetic information. The smaller DNA pieces are called *oligo–nucleotides*. Energetically favorable, if *dA* bonds to *dT* and *dC* pairs with *dG*, which means there exists a one to one correspondence in the set of DNA fragments. There is another important bijective function f: DNA –> RNA, which is based on similar chemically preferred pairs, *dArU*, *dTrA*, *dCrG* and *dGrC* where RNA fragments are variations with repetitive *rA, rU, rC* and *rG* elements. *The Central Dogma of Molecular Biology*, which is the law of genetic information storage and transfer in living organisms is based on these special relationships (Figure 1). Indeed, if one investigate a *sample* using a *DNA–chip* then only the unique complementary basis sequences ought to form *double–helixes*. This process is known as *hybridization*. Nevertheless, every molecular process observed at macroscopic level is stochastic. Depending on the basis sequences it is possible to arise some non exactly complementary distorted *double–helix*. The energetically less stable basis pairs are the so called *mismatches* (*MM*). *MM*s and mutations are closely related, *MM*s can alter into mutations and with the help of them even a single mutation can be detected. Since we can select the probe sequences, it is theoretically possible to indicate all well–known and unknown mutations in the sample. DNA–chip technology takes the advantages of parallel experiments, even up to 100,000 different oligo–nucleotides can be applied on a single chip and the investigation takes only a few minutes. Based on the application area chips can be divided into three major categories:

– *Diagnostic chips (oligoDNA–Chips)*: The length (number of bases) of the used probe oligo–nucleotides is between 20 and 60. They are mainly used in the diagnostics of pathogens and detection of genetically originated disorders.
– *Transcriptional chips (cDNA–Chips)*: The length of the used probe oligo–nucleotides is typically larger than 100 nucleotides. They are often used in cancer research to detect changes and similarities between healthy and tumour cells.
– *Chips for sequence analysis*: The probe nucleotides are quite short. The goal is to determine the overlapping frames and substrings of genes. In fact, sequence analysis is an application of the well–known shortest common superstring problem.

All three kinds of DNA–chips serve to detect the presence or absence of certain nucleotide chains in the analyzed sample. Although the philosophy is basically the same, there are some substantial differences among the groups. In our point of view the most remarkable one is the role of mutations, which is the most important in the first group.

There are several different DNA–chip manufacturing techniques have been developed during the last decade. In the case of diagnostic chips, the most frequently applied procedure is very similar to those used in computer chip fabrication [1]. Photolitho-

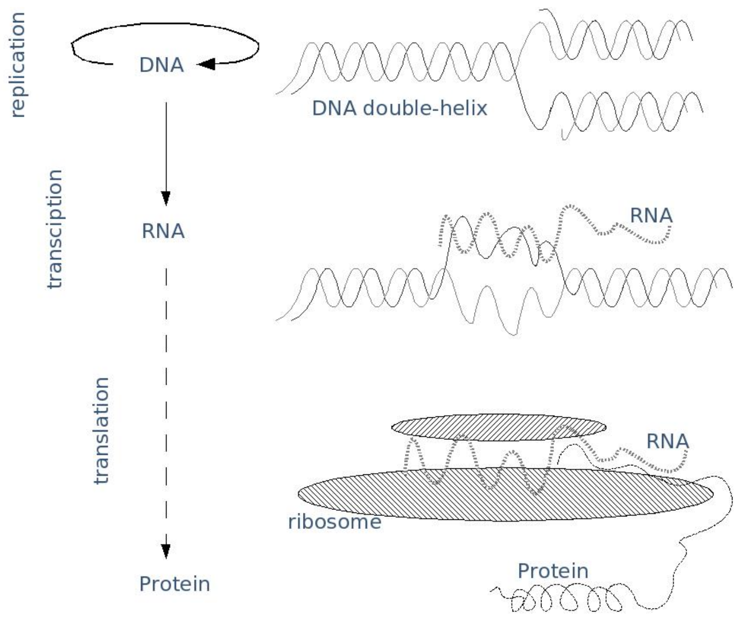


Fig. 1. The flow of genetic information

graphic processes such as photosensitive masks are combined with solid phase chemistry to bond DNA fragments onto the surface. With the series of specific masks and chemical steps high density chips can be constructed. In the development, application and propagation of DNA–chips S. Fodor played a decisive role [2, 3, 4].

2.1 Sample Analysis Using DNA–Chips

The investigation is a multi–step experiment involving the following processes (Figure 2). (i) *Amplification* (multiplication) of DNA strands in the sample to obtain appropriate amount of them. (ii) Labeling of the fragments with flourescent dye to monitor the sample. (iii) Hybridization between the sample and probe sequences immobilized onto the chip surface. (iv) Measuring the fluorescence of labeled DNA fragments to determine where hybridization occured. (v) Data interpretation. Beside the last step, which is also a time–consuming and computer demanding process chip design is apparently the most important step which precedes the investigation and determines its success or failure.

2.2 Role of Hybridization – The Nearest Neighbor Method

The computational prediction of the thermodynamics of hybridization plays a pivotal role in chip design. Accordingly, several methods have already been proposed to estimate the *melting point* or *temperature* (T_m) of nucleic acid duplexes. At the melting temperature 50% of the DNA fragment and its perfect complement form duplex, the other half are in free single stranded state due to molecular thermal motion. In fact, the

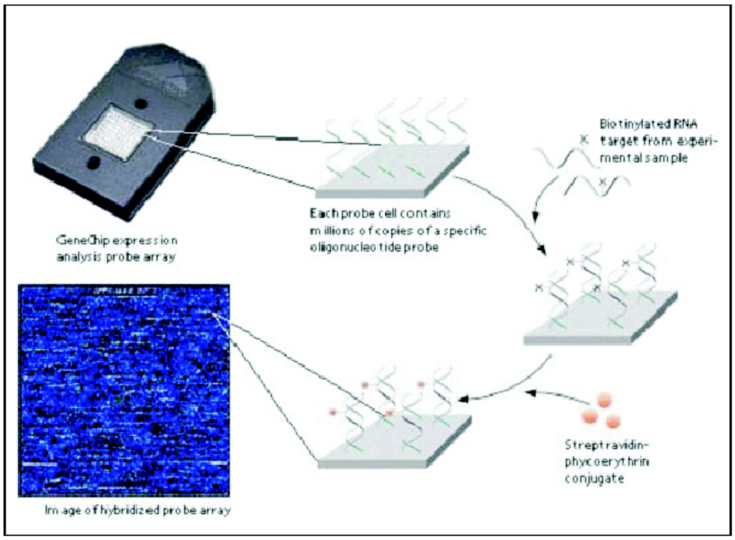

GeneChip expression analysis probe array
Each probe cell contains millions of copies of a specific oligonucleotide probe
Biotinylated RNA target from experimental sample
Streptavidin-phycoerythrin conjugate
Image of hybridized probe array

Fig. 2. The simplified scheme of DNA–chip investigation

probability of hybridization and the T_m–point are descriptors of the same thermodynamical property: the stability of the duplex. The higher the T_m the more likely hybridization occurs. To calculate T_m the simpler methods use the chemical formula of the chains (eq.1.) and empirical parameters are accounted for solvation effects (eq. 2.).

$$T_d = 2(\#A + \#T) + 4(\#C + \#G) \tag{1}$$

where #X means the number of X bases in the sequence.

$$Tm = 8.15 + 16.6 * log[Na^+] + 41(X_G + X_C) - 500/L - 0.62F \tag{2}$$

where L, F are empirical parameters, X_G, X_C can be simply calculated from #G and #C. In the most complicated and flexible case the actual sequence of the chains is also taken into consideration (eq.3.) [5, 6].

$$T_m = \frac{\Delta H_0}{\Delta S_0 + R * ln[c/4]} - 273.15 + const * log[Na^+], \tag{3}$$

where δH_0 is the enthalpy, δS_0 the entropy and c the oligo concentration. δH_0 and δS_0 depend on the base–sequences and 12 constants (*const*) stated by the examination [7, 8].

Although, there is a scientific dispute [9, 10, 11, 12] about the best parameter set the *nearest neighbor* (*NN*) method (eq. 3.) is generally considered to be the most accurate prediction of T_m. Nevertheless, substantial problems have been left unsolved considering *NN*. First of all, the *NN* parameters are not valid in solid phase, they are based on fluid phase measurements. Secondly, they originally were developed to describe the thermodynamics of perfect–matching sequences and their extension to other cases is painful, there still exist quite a few mismatches without parameter. Lastly, every parameter defined by experiments has limited scope. Reparametrization can help to solve these problems, but it requires a tremendous amount of experimental work regarding the *NN* approach. In addition, if one consider carefully the last argument, then this is not else than only a stone on Sisyphus's way. A more effective approach will be presented in the following sections to avoid these difficulties.

2.3 Simulated Annealing

Simulated Annealing (SA, [13]) is a randomized procedure, which supplies good approximation solutions for combinatorial optimization problems in many practical cases [14, 15, 16, 17]. This technology was developed at the beginning of the 80's. As its name implies, the SA exploits an analogy between the way how a metal cools and freezes into a minimum energy crystal structure (the annealing process) and the search for a minimum in a more general system. The success of the process depends strongly on the choice of a control parameter, called temperature. In order to be able to provide as good as possible cooling plan for the Simulated Annealing, the investigation of the procedure's convergence behavior is necessary.

Simulated Annealing is started with a feasible solution of the combinatorial optimization problem and in every iteration a randomly selected neighbour solution is produced. The algorithm employs a random search which not only accepts changes that decrease objective function, but also some changes that increase it. If the change has a better function value, one turns into it and iterates. Otherwise one accepts the new solution only with a certain probability. This probability decreases with increasing the iteration number, because of the decreasing temperature.

3 *In situ, in silico* Chip Design

As we mentioned earlier, regarding DNA the most stable state if the four bases can form Watson–Crick basis pairs: $dG \equiv dC$, $dA = dT$ (where every hyphen means one hydrogen bond). If two sequences are exactly complementary then they will hybridize with each other under appropriate conditions. Since, duplex formation is a stochastic process hybridization can occur between non perfectly matching sequences and its probability is in inverse proportion to the number of *MM*s. Furthermore, one can conclude that different *MM*s might have different effects on hybridization. Accordingly, the following parameters are specified in our model:

1. *Type–dependent parameters*: In the case of DNA, the number of Watson–Crick pairs and *MM*s is 2 and 8, respectively. The number of possible combinations are $\binom{4+2-1}{2}$ (Table 1. a). There are 4 Watson–Crick pairs and 12 *MM*s considering DNA–RNA hybrids, where the order of elements is significant $\binom{4}{1} \times \binom{4}{1}$ (Table 1. b).

   Every type–dependent parameter is restricted into the [0.0, 10.0] interval:

   $0.0 \leq dXrY \leq 10.0$, where $X \in \{A,G,C,T\}$ and $Y \in \{A,G,C,U\}$

   Apparently, the positions of *MM*s in the sequence are not equivalents (Figure 3). That is why, the following parameter type was introduced:

Table 1. a) The commutative Cayley table of DNA/DNA–pairs, the off diagonal elements are *MM*s and the different ones are in italic; b) The Cayley table of DNA/RNA–pairs, the off diagonal elements are *MM*s

dA	dC	dG	dT	DNA/DNA
dAdA	dCdA	dGdA	**dTdA**	dA
dAdC	dCdC	**dGdC**	*dTdC*	dC
dAdG	**dCdG**	*dGdG*	*dTdG*	dG
dAdT	*dCdT*	*dGdT*	*dTdT*	dT

a)

dA	dC	dG	dT	DNA/RNA
dArA	dCrA	dGrA	**dTrA**	rA
dArC	dCrC	**dGrC**	dTrC	rC
dArG	**dCrG**	dGrG	dTrG	rG
dArU	dCrU	dGrU	dTrU	rU

b)

2. *Position–dependent parameters*: they are determined by the length of the sequence and the position of the mismatch:
   - The *length of sequences* is also important; a mismatch has greater importance, if the length of the sequence is shorter. The $f(x)$ function for weighting the position of the mismatch has to be defined according to the length of sequences.
   - The importance of the sequence positions follows a maximum curve regarding *MM*s. The speed of growth or the decrease depends on the length of the sequence and *MM*s have less influence on the probability of hybridization at both ends of the chain, as if they were in the center of the sequence (Figure 3).

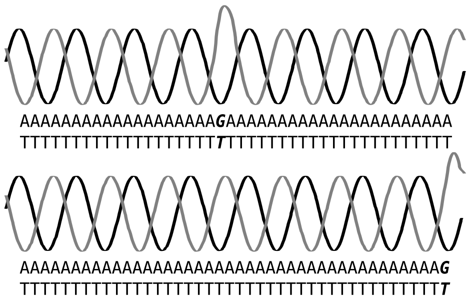

AAAAAAAAAAAAAAAAAAAGAAAAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAG
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

Fig. 3. The influence of the *MM* position on hybridization

3. *Solution–dependent parameters*:

   They cover the experimental conditions i.e. salt and *sample* concentrations and other environmental factors, which also effect the probability of hybridization. The experimental conditions are specified by the scientist, who plans and accomplishes the work. One can assume that these values do not change in a certain laboratory. Although we do not use these parameters explicitly, they are originally involved in our approach.

In our basic model, we optimize only the type–dependent parameters, and set the other parameters to appropriate values (Section 4.2).

4 Methods

All the above mentioned parameters represent the chemical sense in our model. With the help of *MM*s, which are weighted by the appropriate parameters, the mathematical model for the estimation of hybridization probability can be presented:

$$P(hybridization) = max\left\{0, 1 - \left(\sum_{i=0}^{l-1} w_{pos}(i) * w_{mm}(m(i))\right)\right\},$$

where l is the length of sequence, i is the position number, $m(i)$ is the type of mismatch at position i, $w_{pos}(i) \in R$ is the weight of position i and $w_{mm}(m(i))$ is the weight of mismatch type at position i.

If there is no mismatch at the position i, then $w_{pos}(i) = w_{mm}(m(i)) = 0$. If appropriate values are assigned to the parameters, experiments can be replaced by computations.

4.1 Optimization of Parameters

The following target function was used in the optimization process.

$$z = min\left(\sum_{i,j}^{n,m} \left|a_{ij} - b_{ij}\right|^2\right)$$

where the element a_{ij} was derived from the chip experiment (TIFF image processing) and its value based on the hybridization of the row i and column j DNA fragment, n, m are the dimensions of the chip. The b_{ij} elements of the B matrix are computed as follows:

$$b_{ij} = f(ij) = \sum_{k=0}^{l} w_{pos}(k) w_{mm}(S_{ijk}),$$

where l is the length of sequence, S_{ij} is the sequence on the chip in row i and column j, S_{ijk} is the *MM* at position k in this sequence. Thus the elements of matrix B are calculated from character strings, which consist of the variations of the four indications (A, C, G, T).

In a real dataset, the proportions of *MM*s are usually not balanced. In order to proportionately optimize the *MM* weights, we need to scale the number of the present *MM*s and expand the target function. The percent of every mismatch in the dataset is calculated as follows:

$$p_{m(t)} = \frac{N_{m(t)} * 100}{\sum_{t=1}^{12} N_{m(t)}}\%$$

where $m(t)$ is the mismatch, $N_{m(t)}$ is the number of $m(t)$ mismatch in the real dataset. In order to scale the number of *MM*s considering DNA-RNA chips, the proportion of $\frac{100\%}{12} = 8.3333\%$ (the evenly distributed MMs) and $p_{m(t)}$ is taken, thus the scale weight for mismatch $m(t)$ is:

$$sc_{m(t)} = \frac{8.3333}{p_{m(t)}}\%$$

The target function can be extended as follows:

$$z = min\left(\sum_{i,j}^{n,m} \left|a_{ij} - b_{ij}\right|^2 \prod_{k=1}^{l} sc_{m(k)_{ij}}\right)$$

where $sc_{m(k)_{ij}}$ is the scale weight of the mismatch at position k in probe–sequence at position (i, j) on the chip.

4.2 The Setting of Weight Parameters

The position–dependent parameters were not allowed to change during the optimization and a general function was used, which means only the length dependence of *MM*s were taken into consideration. The function used by us is the following:

$$f(i) = \sqrt{\frac{min\left\{dist_{begin}(i), dist_{end}(i)\right\}}{\frac{l}{2}}},$$

where $dist_{begin}(i)$ is the distance of position i from the beginning of sequence, $dist_{end}(i)$ is the distance of position i from the end of sequence and l is the length of the sequence. Figure 4 shows this function.

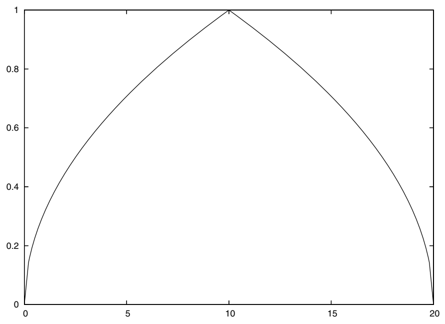


Fig. 4. The weight function for calculating the position–dependent parameters

4.3 Optimization with Simulated Annealing

The search space consists of vectors of the 12 MM parameters. The parameters can have all the values from the $[0.0, 10.0]$ interval. At the beginning of the optimization process, the parameter values are set to 1.0. For the generation of new solutions the values of current parameters are increased with a random number from the interval of $[-0.1, 0.1]$. With the changed parameters the difference of the theoretical and the experimental matrices is computed and compared with the previous result. Based on the Simulated Annealing method these results are taken or rejected.

In order to determine the initial test temperature and the maximum iteration number, the efficiency of the optimization was tested in several intervals. In this study, only the linear cooling schedule was used.

5 Results

Figure 5 shows the differences between average optimum values in the dependence of the initial test temperature and the number of iterations. The initial test temperature was increased from 50 to 140 and number of iterations from 10 to 55. It can be seen that the optimum values increase, if the temperature is between 50 and 70 and the number of iterations are between 50 and 70. In contrast if the temperature is between 100 and 140 and the number of iterations is 20, the difference between the model and experiment is acceptable. Since SA is a stochastic search method, we repeated the samples 100 times for each case.

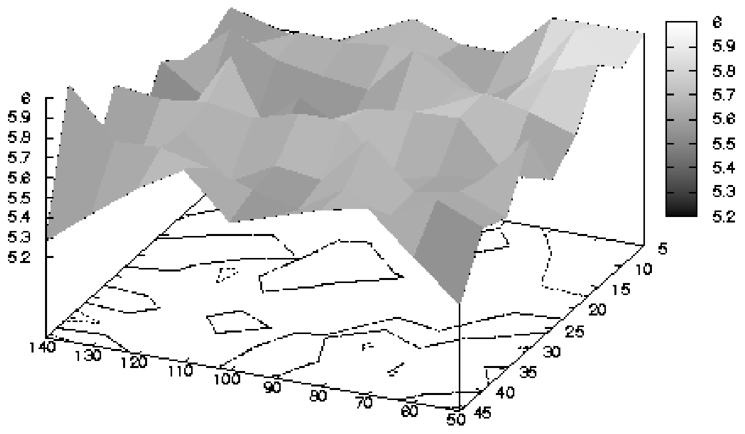


Fig. 5. The average optimum values generated by the method using different temperature and iteration number

5.1 The Comparison of the Optimized Parameter

In Figure 6 and 7 the variation of the parameter values can be observed. Those parameters, which are important from the biochemical point of view are represented by grey columns, the others are in black.

It can be seen that starting form 1.0 at an early stage the values are mixed. However, with the advance of the optimization process the important and the less notable parameters separate from each other and the most important ones obtain the highest weight, eventually. If we take into account the fact that the hybrid RNA/DNA structures are more stable then the DNA/DNA duplex ones and the base pair cytosine and guanine stands for a stronger interaction than the double hydrogen–bonded adenine and thymine (or uracil) the following conclusion can be made:

Regarding DNA/RNA hybrid systems, the RNA–side mismatches should determine mainly the stability of the hybridization. The theoretical consideration stays in coherence with the parameters resulted from the optimization process. As you can see in Figure 7, 5 out of the 6 possible mismatches on the RNA–side possess the first 5 positions based on the weight (dTrC, dArC, dGrG, dArG, dTrG).

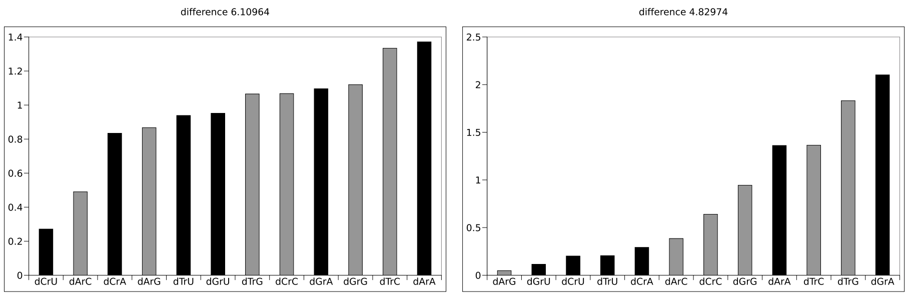


Fig. 6. The values of the type–dependent parameters in the case of goal function 6.10964 and 4.82974

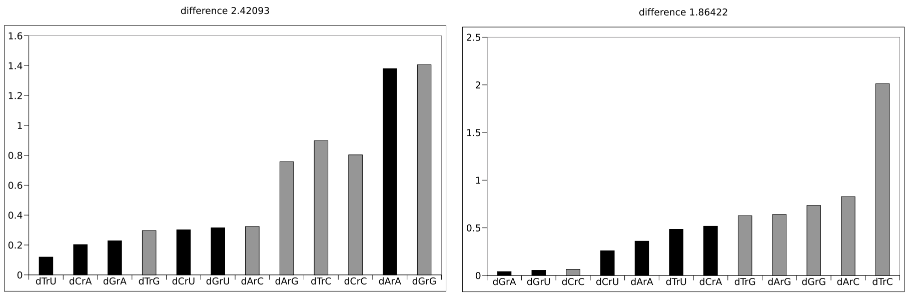


Fig. 7. The values of the type–depended parameters in the case of goal function 2.42093 and 1.86422

6 Conclusion

The prediction of thermodynamical properties of nucleic acids using computer modeling has not been solved yet. The main problem sources are (i) the parameters used in the computation and determined by time consuming and expensive experiments can be applied only to fluid phase, (ii) the lack of *MM* parameters, (iii) the parameters strongly depend on the experimental conditions (e.g. temperature, solvent, etc.).

We presented a novel theoretical model (*in situ in silico approach*) to estimate the hybridization process between DNA/DNA and DNA/RNA strands and eliminate the previously mentioned defects. With the help of this new method, the *in silico* optimization process takes place *in situ* the DNA–chip laboratory, then using the established parameters one can model the hybridization, which is the cornerstone of DNA–chip design.

By the computation done so far the implemented simulated annealing method was used with linear cooling curve. Beside the experimental test of the method, the exponential and Boltzmann–sigmoid cooling scheme are in progress as well as the optimization of the position–dependent parameters.

References

1. Chiu, G.T., Shaw, J.: Optical lithography. In: IBM Journal of Research and Development, Vol.41. (24. Jan. 1997)
2. Chee, M., Yang, R., Hubbel, E., Berno, A., Huang, X., Stern, D., Winkler, J., Lockad, D., .Morris, M., SP.Fodor: Accessing genetic information with high-density dna arrays. In: Science, Vol.274. (25. Oct. 1996) 610–613
3. Fodor, S., Rava, R., Huang, X., Pease, A., Holmes, C., Adams, C.: Multiplexed biochemical assays with biological chips. In: Nature, Vol.364. (5. Aug. 1993) 555–556
4. Fodor, S., Read, J., Pissung, M., Stryer, L., Lu, A., Solas, D.: Light-directed, spatially addressable parallel chemical synthesis. In: Science, Vol.251,No.4995. (15. Feb. 1991) 767–773
5. Panjkovich, A., Melo, F.: Comparison of different melting temperature calculation methods for short dna sequences. In: Bioinformatics. (March 15, 2005) 21(6): 711 – 722
6. Panjkovich, A., Norambuena, T., Melo, F.: dnamate: a consensus melting temperature prediction server for short dna sequences. In: Nucleic Acids Res. (July 1, 2005) 33(suppl_2): W570 – W572
7. Lee, I., Dombkowski, A., Athey, B.: Guidelines for incorporating non-perfectly matched oligonucleotides into target-specific hybridisation probes for a dna microarray. In: Nucleic Acids Research, Vol. 32. No. 2. (2004) 681–690
8. Rouillard, J.M., Zuker, M., Gulari, E.: Oligoarray 2.0: design of oligonucleotide probes for dna microarrays using a thermodinamic approach. In: Nucleic Acids Research, Vol. 31. No. 12. (2003) 3057–3062
9. Breslauer, K., Frank, R., Blocker, H., Marky, L.: Predicting dna duplex stability from the base sequence. In: Proc. Natl. Acad. Sci. USA 83. (1986) 3746–3750
10. Freier, S., R.Kierzek, Jaeger, J., Sugimoto, N., Caruthers, M., Neilson, T., Turner, D.: Improved parameters for predictions of rna rna duplex stability. In: Proc. Natl. Acad. Sci. 83. (1986) 9373–9377
11. Wallace, R., Shaffer, J., Murphy, R., Bonner, J., Hirose, T., Itakura, K.: Hybridization of synthetic oligodeoxyribonucleotides to phi chi 174 dna: the effect of single base pair mismatch. In: Nucleic Acids Res.6. (1979) 3543–3557
12. Howley, P., Israel, M., Law, M., Martin, M.: A rapid method for detecting and mapping homology between heterologous dnas. evaluation of polyomavirus genomes. In: JBC, Vol. 254. (1979) 4876–4883
13. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. In: Science, Vol.220,No.4598. (May 1983) 671–680
14. Berendsen, H., van der Spoel, D., van Drunen, R.: Gromacs: A message passing parallel md implementation. In: Comp. Phys. Comm. 91. (1995) 43–56
15. Sanbonmatsu, K.Y., Joseph, S., Tung, C.S.: Simulating movement of trna into the ribosome during decoding. In: PNAS 2005 102. (2005) 15854–15859
16. Inc., A.S.: Insight ii molecular modeling and simulation enviornment (2005)
17. for Macromolecular Modeling, N.R., Bioinformatics: Scalable molecular dynamics (2005)

A Hybrid GA/SVM Approach for Gene Selection and Classification of Microarray Data

Edmundo Bonilla Huerta, Béatrice Duval, and Jin-Kao Hao

LERIA, Université d'Angers,
2 Boulevard Lavoisier, 49045 Angers, France
{edbonn, bd, hao}@info.univ-angers.fr

Abstract. We propose a Genetic Algorithm (GA) approach combined with Support Vector Machines (SVM) for the classification of high dimensional Microarray data. This approach is associated to a fuzzy logic based pre-filtering technique. The GA is used to evolve gene subsets whose fitness is evaluated by a SVM classifier. Using archive records of "good" gene subsets, a frequency based technique is introduced to identify the most informative genes. Our approach is assessed on two well-known cancer datasets and shows competitive results with six existing methods.

Keywords: Genetic algorithms, Fuzzy logic, Support vector machines, Feature selection, Classification, Microarray data.

1 Introduction

The DNA Microarray technology allows measuring simultaneously the expression level of a great number of genes in tissue samples. A number of works have studied classification methods in order to recognize cancerous and normal tissues by analyzing Microarray data [1, 8, 2]. The Microarray technology typically produces large datasets with expression values for thousands of genes (2000∼20000) in a cell mixture, but only few samples are available (20∼80).

From the classification point of view, it is well known that, when the number of samples is much smaller than the number of features, classification methods may lead to data overfitting, meaning that one can easily find a decision function that correctly classifies the training data but this function may behave very poorly on the test data. Moreover, data with a high number of features require inevitably large processing time. So, for analyzing Microarray data, it is necessary to reduce the data dimensionality by selecting a subset of genes that are relevant for classification.

In the last years, many approaches, in particular various Genetic Algorithms (GAs) and Support Vector Machines (SVMs), have been successfully applied to Microarray data analysis [6, 19, 16, 10, 15, 17, 18, 13]. In Section 3, we review some of the most popular approaches.

In this paper, we are interested in gene selection and classification of DNA Microarray data in order to distinguish tumor samples from normal ones. For this purpose, we propose a hybrid model that uses several complementary techniques: fuzzy logic, a Genetic algorithm (GA) combined with a Support Vector Machine (SVM) and an archive-based gene selection technique. Comparing with previous studies, our approach has several particular features. First, to cope with the difficulty related to high dimensional data, we introduce a fuzzy logic based pre-processing tool which allows to reduce largely the data dimensionality by grouping similar genes. Second, our GA uses archives to record high quality solutions. These archives are then analyzed to identify the most frequently appearing genes which would correspond to the most predictive genes. Third, the GA combined with a SVM classifier is used both for selecting predictive genes and for final gene selection and classification.

The proposed approach is experimentally assessed on two well-known cancer datasets (Leukemia [8] and Colon [1]). Comparisons with six state-of-the-art methods show competitive results according to the conventional criteria.

The remainder of this paper is organized as follows. In Section 2, we describe briefly the two Microarray datasets used in this study. In Section 3, we review some popular gene selection approaches for the classification of Microarray data. In Section 4, we introduce the general scheme of our hybrid model. In Section 5, we describe our GA/SVM approach. Experimental results are presented in Section 6. Finally conclusions are given in Section 7.

2 Datasets

In this study, we use two well-known public datasets, the Leukemia dataset and the Colon cancer dataset. All samples were measured using high-density oligonucleotide arrays [2].

The Leukemia dataset[1] consists of 72 Microarray experiments (samples) with 7129 gene expression levels. The problem is to distinguish between two types of Leukemia, Acute Myeloid Leukemia (AML) and Acute Lymphoblastic Leukemia (ALL). The complete dataset contains 25 AML samples of and 47 ALL samples. As in other experiments [8], 38 out of 72 samples are used as training data (27 ALL samples and 11 AML samples) and the remaining samples (20 ALL samples and 14 AML samples) are used as test data.

The Colon cancer dataset[2] contains the expression of 6000 genes with 62 cell samples taken from colon cancer patients, but only 2000 genes were selected based on the confidence in the measured expression levels [1]. 40 of 62 samples are tumor samples and the remaining samples (22 of 62) are normal ones. In this paper, the first 31 out of 62 samples were used as training data and the remainder samples as test data.

[1] Available at: http://www.broad.mit.edu/cgi-bin/cancer/publications/.
[2] Available at: http://microarray.princeton.edu/oncology/affydata/index.html.

3 Review of Feature Selection Approaches

Feature selection for classification is a very active research topic since many application areas involve data with tens of thousands of variables [9]. This section concerns more specifically a literature review of previous studies on feature selection and classification of Microarray Data, with a special focus on the Leukemia and the Colon datasets presented in Section 2.

Feature selection can be seen as a typical combinatorial problem. Informally, given a dataset described by a large number of features, the aim is to find out, within the space of feature subsets, the smallest subset that leads to the highest rate of correct classification. Given the importance of feature selection, many solution methods have been developed. Roughly speaking, existing methods for feature selection belong to three main families [9]: the filter approach, the wrapper approach and the embedded approach.

The filter methods separate the feature selection process from the classification process. These methods select feature subsets independently of the learning algorithm that is used for classification. In most cases, the selection relies on an individual evaluation of each feature [8, 6], therefore the interactions between features are not taken into account.

In contrast, the wrapper approach relies on a classification algorithm that is used as a black box to evaluate each candidate subset of features; the quality of a candidate subset is given by the performance of the classifier obtained on the training data. Wrapper methods are generally computation intensive since the classifier must be trained for each candidate subset. Several strategies can be considered to explore the space of possible subsets. In particular, in [14], evolutionary algorithms are used with a k-nearest neighbor classifier. In [12], the author develops parallel genetic algorithms using adaptive operators. In [18], one finds a SVM wrapper with a standard GA. In [20], the selection-classification problem is treated as a multi-objective optimization problem, minimizing simultaneously the number of genes (features) and the number of misclassified examples.

Finally, in embedded methods, the process of selection is performed during the training of a specific learning machine. A representative work of this approach is the method that uses support vector machines with recursive feature elimination (SVM/RFE) [10]. The selection is based on a ranking of the genes and, at each step, the gene with the smallest ranking criterion is eliminated. The ranking criterion is obtained from the weights of a SVM trained on the current set of genes. In this sense, embedded methods are an extension of the wrapper models. There are other variants of these approaches, see [21, 7] for two examples.

4 General Model for Gene Selection and Classification

The work reported in this paper is based on a hybrid approach combining fuzzy logic, GA and SVM. Our general model may be characterized as a three-stage sequential process, using complementary techniques to shrink (or reduce) gradually the search space. The rest of this section gives a brief description of these three stages.

Stage 1 ***Pre-processing by fuzzy logic.*** This stage aims to reduce the dimension of the initial problem by eliminating gene redundancy. This stage is basically composed of four steps. First, the gene expression levels are transformed into fuzzy subsets with Gaussian representations. Second, the Cosine amplitude method is employed to assess fuzzy similarities between genes. We build a similarity matrix that is then transformed to a matrix of fuzzy equivalence relations by different compositions. Third, using $\alpha-$cuts [23] with decreasing values of α, we obtain groups of similar genes that correspond to fuzzy equivalence classes of genes. Fourth, for each group, one gene is randomly taken as the representative of the group and other genes of the group are ignored. Applying this dimension reduction technique to the datasets presented in Section 2, the set of 7129 genes for Leukemia (2000 genes for Colon respectively) is reduced to 1360 genes (943 genes respectively). Therefore, the search space is dramatically reduced. As we show later in Section 6, with this reduced set of genes, we will be able to obtain high quality classification results. A detailed description of this stage goes beyond the scope of this paper and can be found in [3].

Stage 2 ***Gene subset selection by GA/SVM.*** From the reduced set of genes obtained in the previous pre-processing stage, this second stage uses a wrapper approach that combines a GA and a SVM to accomplish the feature (gene) subset selection. The basic idea here consists in using a GA to discover "good" subsets of genes, the goodness of a subset being evaluated by a SVM classifier

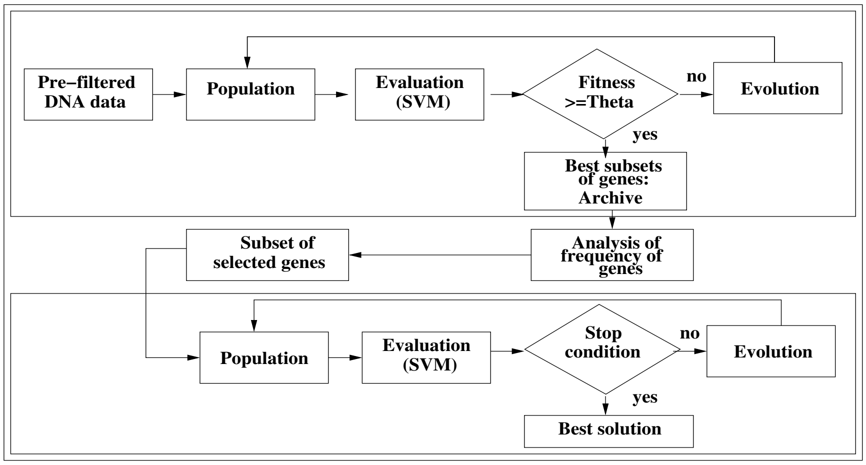


Fig. 1. The general process for gene subset selection and classification using GA/SVM: Gene subset selection (Stage 2 - top); Gene selection and classification (Stage 3 - bottom)

on a set of *training data* (see Section 2). During this stage, high quality gene subsets are recorded to an archive in order to be further analyzed.

At the end of the GA, the analysis of the archived gene subsets is performed: gene subsets are compared among them and the most frequently appearing genes are identified. This process typically leads to a further reduced set of genes (<100 genes for the Leukemia and Colon dataset). Fig.1 (top) shows a general picture of this stage.

Stage 3 *Classification.* Stage 2 has identified a reduced set of relevant genes which is now used in the final step of gene selection and classification. From this set of genes, a new round of search is carried out using the previous GA/SVM, this time to classify the *test data* (see Section 2). This stage will thus select the most predictive genes to classify the test data. Fig.1 (bottom) shows a general picture of this stage.

5 Gene Selection and Classification by GA/SVM

We describe now the hybrid GA/SVM algorithm for carrying out Stages 2 and 3 of the general model for gene selection and classification. As explained previously, the GA is designed both for discovering good gene subsets and for final gene selection and classification. The SVM-based classifier is used to ensure the fitness evaluation of each candidate gene subset. One important feature of the GA developed in this work is the use of an archive to record quality gene subsets discovered during the gene subset selection stage. This archive is then analyzed to identify a small number of highly frequently appearing genes that are used in the final classification stage. Notice that the idea of archiving good solutions is not really a new one because it is already used in some multiobjective evolutionary algorithms [26]. However, as we will see later in Section 5.3, our way of exploiting the information of the archive to identify predictive genes is original and useful.

From these retained genes obtained from archive analysis, the same GA/-SVM algorithm is applied to the test data to perform the final gene selection and classification tasks.

5.1 The Genetic Algorithm

General Schema. The basic components of our GA are presented later in this section. Here we show the general algorithm. The GA follows a generational schema with a form of elitism. To obtain a new population from the current population P, the top E% of the population P are recorded, E being fixed to 10% or 15% in our experiments (see Section 6). Then, the following two actions are taken: 1) select two parents and apply (with a given probability) the crossover to create two new solutions which are muted (with a given probability), and 2) replace the parents by their offspring. These two actions are repeated for a pre-fixed number of times. Finally, the recorded elite chromosomes are copied backed to the population P to replace the worst rated chromosomes. At this point, one generation is accomplished.

Chromosome and initial population. The chromosomes are binary-encoded, each allele (bit) of the chromosome represents a gene. If an allele is '1' it means that this gene is kept in the gene subset and '0' indicates that the gene is not included in the subset. Each chromosome represents thus a gene subset. For Stage 2 of the general model, the chromosome length is equal to the number of genes pre-selected by the fuzzy pre-processing (i.e. 1360 for the Leukemia dataset and 943 genes for the Colon dataset). For Stage 3, the chromosome length depends on the size of the gene subset retained after analyzing the solution archive (see section 5.3). In both cases, the initial population of the GA is randomly generated according to a uniform distribution.

Fitness function. The fitness of a chromosome, i.e. a subset of genes, is assessed by the classification rate on the initial datasets. In other words, a subset of genes leading to a high classification rate is considered to be better than a subset leading to a low classification rate. In our case, a SVM classifier (see Section 5.2) ensures this classification task.

Selection, crossover, mutation, and replacement. We use the roulette wheel selection and random one-point crossover and multi-uniform mutation operators. Offspring replaces always their parents. An elitism mechanism is also applied to conserve the top 10% or 15% chromosomes of the population between two successive generations.

Archives of high quality gene subsets. Given a chromosome (a candidate subset of genes), the SVM classifier gives its fitness in terms of classification rate on the training data set. If the classification rate is high enough (defined by a threshold theta, see Fig. 1.a), the subset of genes is recorded in an archive. In this paper, the threshold theta is set to 0.90 and 0.91 respectively for the Leukemia and Colon dataset.

Stopping criterion. The evolution process ends when a pre-defined number of generations is reached or a fitness value of 100% is obtained.

5.2 The SVM Classifier

Support Vector Machines [24] are basically binary classification algorithms. When the data are linearly separable, SVM computes the hyperplane that maximizes the margin between the training examples and the class boundary. When the data are not linearly separable, the examples are mapped to a high dimensional space where such a separating hyperplane can be found. The mechanism that defines this mapping process is called the kernel function. SVM are powerful classifiers with good performance in the domain of Microarray data [10, 17]. They can be applied to data with a great number of genes, but it has been showed that their performance is increased by reducing the number of genes [6, 2].

In our wrapper GA/SVM algorithm, we use a SVM classifier to assess the quality of a gene subset. For a chromosome x that represents a gene subset, we apply a Leave-One-Out Cross-Validation (LOOCV) method to calculate the average accuracy (rate of correct classification) of a SVM trained with this gene subset [11]. The LOOCV procedure means that one sample from the dataset is considered as a test case while a SVM is trained on all the other samples, and this evaluation is repeated for each sample. So for each chromosome x, $Fitness(x) = accuracy_{SVM}(x)$.

One of the key elements of a SVM classifier concerns the choice of its kernel. In our study, we have chosen to use the RBF kernel. We also experimented Gaussian and polynomial kernels. For polynomial kernels, the main difficulty is to determine an appropriate polynomial degree while the results we obtained with the Gaussian kernel are not satisfactory. Notice that RBF has been used in several previous studies for Microarray data classification [4, 18, 5].

5.3 Archive Analysis

At the end of stage 2 and prior to the final classification (Stage 3), the archive is analyzed and the most frequently appearing genes in the archive are retained for the final gene selection and classification (stage 3). Typically, this analysis will lead to a limited number of genes (between 50 to 100). From these genes, the GA/SVM algorithm will then determine the final set of genes relevant to classify the data.

6 Experimental Results and Comparisons

6.1 Parameters Settings

For our GA/SVM algorithm, the GA is implemented in Matlab (Version 5.3.1 for Windows). The SVM classifier is based on the SVM Toolbox developed by Gavin Cawley[3].

Table 1. GA parameters for the stage of gene subset selection (Stage 2)

Parameters	Leukemia	Colon
Size of population	500	500
Length of chromosome	1360	943
Number of generations	2500	2500
Crossover rate	0.95	0.98
Mutation rate	0.02	0.01
Elitism rate E	10%	15%

The GA parameters used in our model of gene subset selection for the Leukemia and Colon datasets are shown in Tables 1 and 2. For the SVM classifier, the same parameters settings are used in the two stages of gene subset selection and classification. The normalization parameter C is fixed at 100 and the control parameter γ for the RBF kernel of SVM is fixed to 0.5. Notice that

[3] http://theoval.sys.eua.uk/~gcc/svm/toolbox

Table 2. GA parameters for the stage of classification (Stage 3)

Parameters	Leukemia	Colon
Size of population	50	50
Length of chromosome	100	50
Number of generations	500	500
Crossover rate	0.985	0.985
Mutation rate	0.02	0.01
Elitism rate E	15%	15%

given the input data used by the GA/SVM are already normalized during the Fuzzy Logic pre-processing, the normalization parameter C has in fact little influence in our case.

6.2 Results and Comparisons

To carry out our experiments, our GA/SVM algorithm is run 5 times on each of the Leukemia and Colon datasets. To calculate the average classification rate of a given gene subset, the LOOCV procedure [11] is employed.

Table 3 summarizes our results (Column 2) for the Leukemia and Colon datasets together with the results of six state-of-the-art methods from the literature (Columns 3-8). The conventional criteria are used to compare the results: the classification accuracy in terms of the rate of correct classification (first number) and the number of used genes (the number in parenthesis, "-" indicating that the number of genes is not available). For AG/SVM, the classification rate that we present is the average classification rate obtained from the 5 independent runs and the number of selected genes is the minimum number obtained from these runs. Detailed results can be found in Table 4.

As it can be observed, for the Leukemia dataset, we obtain a classification rate of 100% using 25 gens, which is much better than that reported in [6, 5]. This same performance is achieved by [25, 18, 20, 10], with fewer genes selected. [20] and [10] reports the minimal number of genes. However, in [20] the evolutionary method begins with a largely reduced set of 50 genes, published in [8] as interesting genes.

The most interesting results that we obtained with our model concern the Colon dataset since our approach offers the highest (averaged) correct classification rate (99.41%); the number of selected genes is greater than the one obtained by [20] or by [25, 10], but it is smaller than the one reported in [18]. An analysis

Table 3. Comparison of GA/SVM with six state of the art methods

	Methods						
Dataset	GA&SVM	[6]	[25]	[18]	[5]	[20]	[10]
Leukemia	100(25)	94.10(-)	100(8)	100(6)	95.0(-)	100(4)	100(2)
Colon	99.41(10)	90.30(-)	91.9(3)	93.55(12)	91.0(-)	97.0(7)	98.0(4)

Table 4. GA/SVM performance on 5 runs

Runs	Run 1	Run 2	Run 3	Run 4	Run5	Average class. rate
Leukemia	100(25)	100(28)	100(30)	100(46)	100(35)	100
Colon	99.64(10)	99.83(15)	97.88(10)	99.83(15)	99.83(15)	99.41

of our results shows that several biologically significant genes reported in [8] are found by our approach.

Table 4 shows the detailed results of 5 independent runs of our GA/SVM algorithm. As it can be observed, these results are quite stable. For the Leukemia dataset, each of the 5 runs obtains a classification rate of 100% while for the Colon dataset, the best run gives a classification rate of 99.64. Even the worst obtains a classification rate of 97.88.

7 Conclusions

In this paper, we presented a general approach for gene selection and classification of high dimensional DNA Microarray data. This approach begins with a fuzzy logic based pre-processing technique that aims to cope with the imprecise nature of the expression levels and to reduce the initial dimension of the input dataset. Following this pre-processing stage, a hybrid wrapper system combining a Genetic Algorithm with a SVM classifier is used to identify potentially predictive gene subsets that are then used to carry out the final gene selection and classification tasks. Another important feature of our approach concerns the introduction of an archive of high quality solutions, which allows limiting the GA/SVM exploration to a set of frequently appearing genes.

This approach was experimentally evaluated on the widely studied Leukemia and Colon cancer datasets and compared with six previous methods. The results show that our approach is able to obtain very high classification accuracy. In particular, to our knowledge, this is the first time that a averaged correct classification rate of 99.41% (with 10 genes) is reached for the Colon dataset.

This approach can be further improved on several aspects. First, we notice that our method does not provide the smallest number of genes on the Leukemia data. This is due to the fact that the GA is only guided by the criterion of classification accuracy. Therefore, the criterion of the number of genes should be integrated into the fitness function. This can be achieved by an aggregated fitness function or a bi-criteria evaluation. Second, the high computation time required in stage 2 can be reduced by the use of a faster classifier (or an approximate fitness function). For example, the m-features operator reported in [22] may be considered. Also, a fine-tuning of SVM parameters in stage 3 may lead to improved results. Finally, we intend to apply our approach to other DNA chip data and to study the behavior of our model.

Acknowledgments. E. Bonilla Huerta is supported by a Mexican CoSNET research scholarship. This work is partially carried out within the French Ouest

Genopole Program. We thank the reviewers of the paper for their very helpful comments.

References

1. U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. In *Proc. Natnl. Acad. Sci. USA*, volume 96, 1999.
2. A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. *Journal of Computational Biology*, 7(3-4):559–583, 2000.
3. E. Bonilla Huerta, B. Duval, and J.K. Hao. Feature space reduction of large scale gene expression data using Fuzzy Logic. Technical Report, LERIA, University of Angers, January 2006.
4. M.P.S. Brown, W.N. Grundy, D. Lin, N. Cristianini, S.W. Sugnet, T.S. Furey, M. Ares Jr., and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines *Proc. Natl. Acad. Sci. U S A.*, 97(1): 262–267, 2000.
5. S. Chao and C. Lihui Feature dimension reduction for microarray data analysis using locally linear embedding. In *APBC*, pages 211–217, 2005.
6. T. S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.
7. L. Goh, Q. Song, and N. Kasabov. A novel feature selection method to improve classification of gene expression data. In Proceedings of the Second Asia-Pacific Conference on Bioinformatics, pages 161–166, Australian Computer Society, Darlinghurst, Australia, 2004.
8. T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
9. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
10. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.
11. T. Joachims Estimating the Generalization Performance of a SVM Efficiently. *Proceedings of the International Conference on Machine Learning (ICML)*, Morgan Kaufman, 2000.
12. L. Jourdan. Metaheuristics for knowledge discovery : Application to genetic data (in French). PhD thesis, University of Lille, 2003.
13. K-J. Kim and S-B. Cho. Prediction of colon cancer using an evolutionary neural network. *Neurocomputing (Special Issue on Bioinformatics)*, 61:361–379, 2004.
14. L. Li, C. R. Weinberg, T.A. Darden, and L.G. Pedersen. Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. *Bioinformatics*, 17(12):1131–1142, 2001.
15. J. Liu and H. Iba. Selecting informative genes using a multiobjective evolutionary algorithm. In *Proc. of Congress on Evolutionary Computation (CEC'02)*, pages 297–302, 2002.

16. F. Markowetz, L. Edler, and M. Vingron. Support vector machines for protein fold class prediction. *Biometrical Journal*, 45(3):377–389, 2003.
17. S. Mukherjee. *Classifying Microarray Data Using Support Vector Machines.* Springer-Verlag, Heidelberg, 2003.
18. S. Peng, Q. Xu, X.B. Ling, X. Peng, W. Du, and L. Chen. Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *FEBS Letter*, 555(2):358–362, 2003.
19. S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J.P. Mesirov, T. Poggio, W. Gerald, M. Loda, E.S. Lander, and T.R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proc. Natl. Acad. Sci. U S A.*, 98(26):15149–15154, 2001.
20. A. R. Reddy and K. Deb. Classification of two-class cancer data reliably using evolutionary algorithms. Technical Report. KanGAL, 2003.
21. Y. Saeys, S. Aeyels Degroeve, D. Rouze, and Y. P. Van de Peer. Feature selection for splice site prediction: A new method using eda-based feature ranking. *BMC Bioinformatics*, 5-64, 2004.
22. S. Salcedo-Sanz, F. Prez-Cruz, G. Campsand, and C. Bousoo-Calzn. Enhancing genetic feature selection through restricted search and Walsh analysis. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34:398–406, 2004.
23. T.J. Ross. *Fuzzy Logic with Engineering Applications.* McGraw-Hill, 1997.
24. V. N. Vapnik. *Statistical Learning Theory.* Wiley N.Y., 1998.
25. Y. Wang, F. Makedon, J.C. Ford, and J.D. Pearlman. Hykgene: a hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data. *Bioinformatics*, 21(8):1530–1537, 2005.
26. E. Zitzlere, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* 8(2): 173-195, 2000.

Multi-stage Evolutionary Algorithms for Efficient Identification of Gene Regulatory Networks

Kee-Young Kim, Dong-Yeon Cho, and Byoung-Tak Zhang

Biointelligence Lab, School of Computer Science and Engineering,
Seoul National University, Seoul 151-742, Korea
{kykim, dycho, btzhang}@bi.snu.ac.kr

Abstract. With the availability of the time series data from the high-throughput technologies, diverse approaches have been proposed to model gene regulatory networks. Compared with others, S-system has the advantage for these tasks in the sense that it can provide both quantitative (structural) and qualitative (dynamical) modeling in one framework. However, it is not easy to identify the structure of the true network since the number of parameters to be estimated is much larger than that of the available data. Moreover, conventional parameter estimation requires the time-consuming numerical integration to reproduce dynamic profiles for the S-system. In this paper, we propose multi-stage evolutionary algorithms to identify gene regulatory networks efficiently. With the symbolic regression by genetic programming (GP), we can evade the numerical integration steps. This is because the estimation of slopes for each time-course data can be obtained from the results of GP. We also develop hybrid evolutionary algorithms and modified fitness evaluation function to identify the structure of gene regulatory networks and to estimate the corresponding parameters at the same time. By applying the proposed method to the identification of an artificial genetic network, we verify its capability of finding the true S-system.

1 Introduction

Although mathematical modeling for the biochemical networks can be achieved at different level of detail (see [1] and [2] for the reviews of metabolic and genetic regulatory networks modeling), we can cluster them into three dominant approaches [3]. One extreme case is mainly intended to describe the pattern of interactions between the components. Graph-based representation gives us the insight for large architectural features within a cell and allows us to discovery principles of cellular organization [4]. However, it is difficult to handle the dynamics of the whole system since these models are very abstract. The other extreme primarily focuses on describing the dynamics of the systems by some kinds of equations which can explain the biochemical interactions with stochastic kinetics [5, 6]. While these approaches lead to realistic, quantitative modeling on cellular dynamics, the application is limited to the small systems due to their computational complexities.

One of the appropriate approaches for the pathway structure and dynamics identification is S-system [7]. It is represented as a system of ordinary differential equations which have a particular form, where each component process is characterized by power-law functions. S-system is not only general enough to represent any nonlinear relationship among the components but also able to be mapped onto the network structure directly. In spite of these advantages, there is a serious drawback which prevents the S-system from wildly spreading over the systems biology communities. Its large number of parameters should be estimated for the small number of observed dynamic trends. Provided that n components such as genes and proteins are involved in a certain living system, we must optimize at least $2n(n+1)$ parameters for the S-system.

Evolutionary computation has been used from its inception for automatic identification of a given system or process [8]. For the S-system models, some evolutionary search techniques have been proposed [9-12]. However, they require the time-consuming numerical integrations to reproduce dynamic profiles for the fitness evaluations. To avoid this problem, Almeida and Voit have employed an artificial neural network (ANN) to smooth the measured data for obtaining slopes of gene expression level curves [13]. By comparing the slope of each S-system in the population with the estimated one from ANN, we can evaluate the fitness values of the individuals without the computationally expensive numerical integrations of differential equations. This method also provides the opportunity for a parallel implementation of the identification task since a tightly coupled system of non-linear differential equations can be separated. Hence, they are able to reduce the time complexity drastically. While collocation method [14] can save the computational cost by approximating dynamic profiles, their estimated systems tend to be invalid since the number of measured data is usually insufficient. This lack of data problem can be resolved by sampling new points from the fitted curves. For the well-estimated profiles, however, we should determine the optimal topology of the artificial neural network such as the number of hidden units and layers.

In this paper, we propose multi-stage evolutionary algorithms to identify gene regulatory networks efficiently. With the symbolic regression by genetic programming (GP), we could evade the numerical integration steps. Here, we have no need to predetermine the topology of the model for the expression profiles since genetic programming can optimized the topology automatically. We also develop hybrid evolutionary algorithms to identify the structure of gene regulatory networks and to estimate the corresponding parameters at the same time. Most previous evolutionary approaches for the S-system identification have used the structural simplification procedure in which some parameters whose values are less than a given threshold are reset to zero. Although this method is able to make the network structure sparse, the true connections which represent somewhat small effect can be deleted during the procedures. That is, it is not easy to set the suitable value for the threshold. In our scheme, Binary matrices for a network structure and real vectors and matrices for parameter values of S-system are combined into a chromosome and co-evolved to find the best descriptive model for the given data. Hence we can identify the S-system without specifying the threshold values for the structural simplification. By applying the proposed method to the artificial gene expression profiles, we successfully identified the true structure and estimated the reasonable parameter values with the smaller number of data than the previous study.

2 Materials and Methods

2.1 S-System

The S-system [7, 15] is a set of nonlinear differential equations described as follows:

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}}, \quad i = 1,2,...,n, \tag{1}$$

where n is the number of dependent variables whose concentrations X_i change dynamically according to above equations and m is the number of independent variables whose concentrations remain constant during the processes. The non-negative parameters α_i and β_i are called rate constants. The real-value exponents g_{ij} and h_{ij} are kinetic orders to represent the interactive effect of X_j to X_i. These differential equations can be divided into two components. The first term represents influences that increase X_i and the second term represents influences that decrease X_i. Thus, X_j induces the synthesis of X_i in case $g_{ij} > 0$, whereas X_j inhibits the increase of X_i if $g_{ij} < 0$. Similarly, a positive (negative) value of h_{ij} indicates that X_j expedites (restrains) the decline of X_i.

We should estimate at least $2n(n+1)$ parameters even if the values related to the independent variables are assumed to be known. That is, α_i, β_i, g_{ij}, and h_{ij} are parameters that must be estimated by evolutionary algorithms (EAs). The generally adopted fitness function for EAs is the sum of relative squared errors:

$$E = \sum_{i=1}^{n} \sum_{t=1}^{T} \left(\frac{X_i(t) - \hat{X}_i(t)}{X_i(t)} \right)^2, \tag{2}$$

where T is the number of sampling points for fitness evaluation, X is the measured data points from the biological experiments and $\hat{X}$ is the values obtained by the numerical integration step of the S-system in the population.

2.2 Symbolic Regression by Genetic Programming

As we mentioned in the introduction, numerical integration steps for the fitness evaluations are very time-consuming. To circumvent these processes, we propose 2-stage evolutionary algorithms for finding the structures and parameters of S-systems. As a preprocessing step, genetic programming (GP) [16] performs symbolic regression for the given time-course data. Through this step, we can predict the dynamic profiles of the given S-system and obtain more data points as well as the derivations of the point for the second stage of our algorithm. This allows us to get rid of the numerical integrations in the fitness evaluations. Compared with the study which employed the artificial neural networks [13], genetic programming has the following advantages for the curve fitting tasks. First, there is no necessity for adjusting the number of hidden layers and units. Second, the results of GPs are more understandable than those of neural networks since GPs return some mathematical functions instead of the illegible graph and a set of weights. Hence, we can easily obtain derivations of each time point if the result functions of GPs are differentiable.

2.3 Hybrid Evolutionary Algorithms

At the second stage of our evolutionary algorithm, we use a hybrid evolutionary algorithm for searching the structures and parameters of S-system. The whole procedure of our algorithm is summarized in Fig. 1.

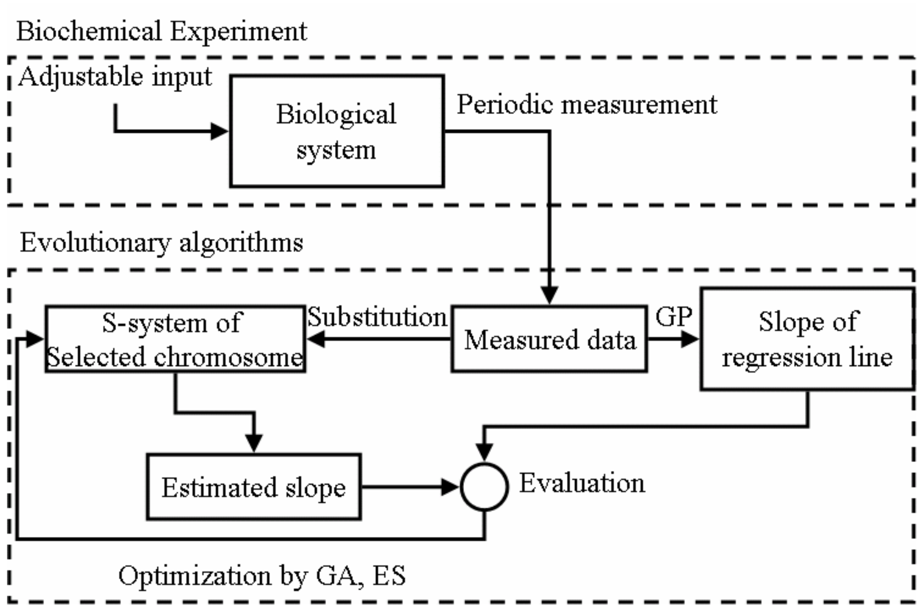
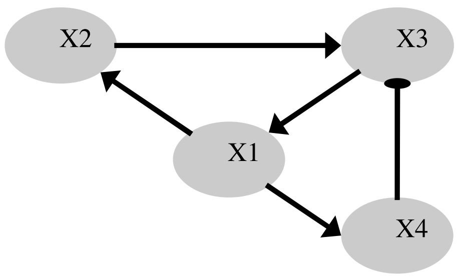


Fig. 1. The whole proposed multi-stage evolutionary algorithm for the identification of S-system

In biochemical experiment step, the dynamic profiles of the involved genes can be obtained by the periodic measurement. We are also able to predict slopes at each time point from the regression line of the measured data by genetic programming. Then, we can evaluate and optimize the chromosomes of evolutionary algorithms by comparing the estimated slopes which came from the data substitution into the S-system with the predicted slopes of the GP regression lines. Hence, the fitness values of each S-system can be evaluated without the time-consuming numerical integrations as follows:

$$E = \sum_{i=1}^{n} \sum_{t=1}^{T} \left(\frac{\dot{X}_i(t) - X_i'(t)}{\dot{X}_i(t)} \right)^2 , \qquad (3)$$



(a) graph representation

$$\frac{dX_1}{dt} = 12X_3^{0.8} - 10X_1^{0.5} \qquad \frac{dX_2}{dt} = 8X_1^{0.5} - 3X_2^{0.75}$$

$$\frac{dX_3}{dt} = 3X_2^{0.75} - 5X_3^{0.5}X_4^{0.2} \qquad \frac{dX_4}{dt} = 2X_1^{0.5} - 6X_4^{0.8}$$

(b) S-system

$$g = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \qquad h = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

binary matrices for the structure

$$\alpha = \begin{pmatrix} 12.0 \\ 8.0 \\ 3.0 \\ 2.0 \end{pmatrix} \quad g = \begin{pmatrix} 0.0 & 0.0 & -0.8 & 0.0 \\ 0.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.75 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.0 & 0.0 \end{pmatrix} \quad \beta = \begin{pmatrix} 10.0 \\ 3.0 \\ 5.0 \\ 6.0 \end{pmatrix} \quad h = \begin{pmatrix} 0.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.75 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.2 \\ 0.0 & 0.0 & 0.0 & 0.8 \end{pmatrix}$$

real vectors and matrices for the parameters

(c) the structure of the chromosome

Fig. 2. (a) The true structure of the artificial gene regulatory network (b) S-system representation (c) The structure of the chromosome in our hybrid evolutionary algorithms

where T is the number of sampling points, $\dot{X}$ is the gradient of the regression line obtained by the genetic programming and X' is the calculated value of each equation in the S-system.

We develop hybrid evolutionary algorithms to identify the structure of gene regulatory networks and to estimate the corresponding parameters at the same time. In this scheme, Binary matrices for a network structure and real vectors and matrices for parameter values of S-system are combined into a chromosome (Fig. 2) and co-evolved to find the best descriptive model for the given data. While crossover operator is applied to binary matrices for searching the structure of the system, their corresponding parameter values also exchanges. This kind of crossover can inherit the good structures as well as the parameter values in the parents to the offspring. That is, we use a row exchange crossover which simply selects the row of the matrix g or h (or both) on the parents, and swaps each other with the parameter values in the real vectors and matrices. For example, Fig. 3(a) shows the case in which we select the second row of *g* on parents. Mutation operator randomly selects an equation of a parent, and then inserts or deletes a component at the selected parent as shown in Fig. 3(b) and (c). These operators perform the local searches in the structure space. In this case, the parameter values are randomly generated for the insertion and reset to zero for the deletion.

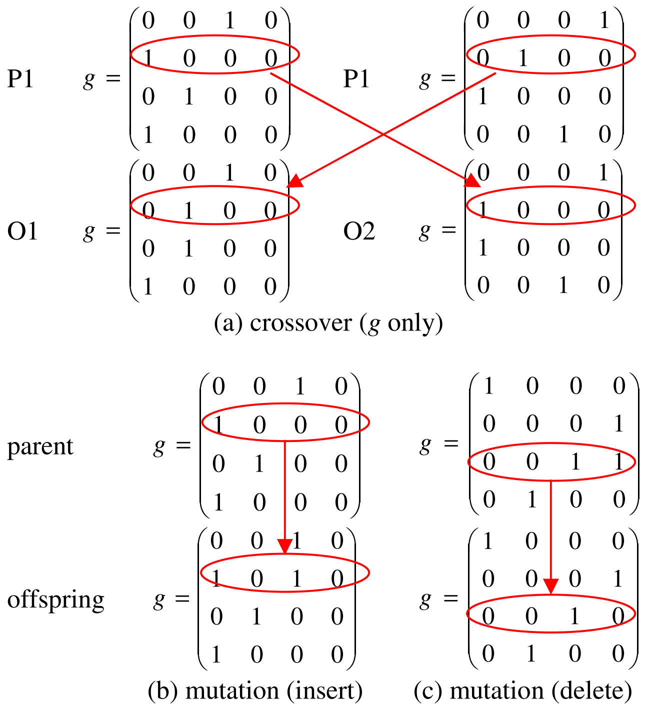

P1
$g =$
$\begin{pmatrix}0&0&1&0\\1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix}$
P1
$g =$
$\begin{pmatrix}0&0&0&1\\0&1&0&0\\1&0&0&0\\0&0&1&0\end{pmatrix}$
O1
$g =$
$\begin{pmatrix}0&0&1&0\\0&1&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix}$
O2
$g =$
$\begin{pmatrix}0&0&0&1\\1&0&0&0\\1&0&0&0\\0&0&1&0\end{pmatrix}$
(a) crossover (g only)
parent
$g =$
$\begin{pmatrix}0&0&1&0\\1&0&0&0\\0&1&0&0\\1&0&0&0\end{pmatrix}$
$g =$
$\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&1\\0&1&0&0\end{pmatrix}$
offspring
$g =$
$\begin{pmatrix}0&0&1&0\\1&0&1&0\\0&1&0&0\\1&0&0&0\end{pmatrix}$
$g =$
$\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix}$
(b) mutation (insert)
(c) mutation (delete)

Fig. 3. Crossover and mutation operators for the binary matrices

We also give some variety to the fitness function in equation (3). In conventional scheme, all points of data have the same weights on the fitness values. However, it is difficult to fit the data points which have large second-order differentiation. Moreover, this makes the parameter values of the chromosomes different from the true one even if they have good fitness values. Thus we multiply the second-order differentiation to each term of evaluation function. The modified fitness function in our algorithm described as follows:

$$E = \sum_{i=1}^{n}\sum_{t=1}^{T}\ddot{X}_i(t)\left(\frac{\dot{X}_i(t) - X_i'(t)}{\dot{X}_i(t)}\right)^2, \tag{4}$$

where $\dot{X}$ is the gradient of the GP regression line, $\ddot{X}$ is second-order differentiation and X' is the calculated value of the each equation in the S-system. By introducing $\ddot{X}$

to fitness function, we can obtain better fitness value of true structure than those of other structures. After the fitness values of the offspring created by the crossover and mutation according to their probabilities are evaluated by equation (4), parameters in the real vectors and matrices are adjusted through the (1+1) evolutionary strategy [17].

We employ the restricted tournament selection (RTS) proposed originally in [18] to prevent the premature convergence on a local-optimum structure and to find multiple topology candidates. In RTS, a subset of the current population is selected for each newly created offspring. The size of these subsets is fixed to some constant called the window size. Then, the new offspring competes with the most similar member of the subset. Since the window size is set to the population size in our implementation, each offspring is compared with all S-system in the current population. If the new one is better, it replaces the corresponding individual; otherwise, the new one is discarded. For the similarity measure, we calculate the structural hamming distances between the new offspring and all individuals in the population by using the binary matrices.

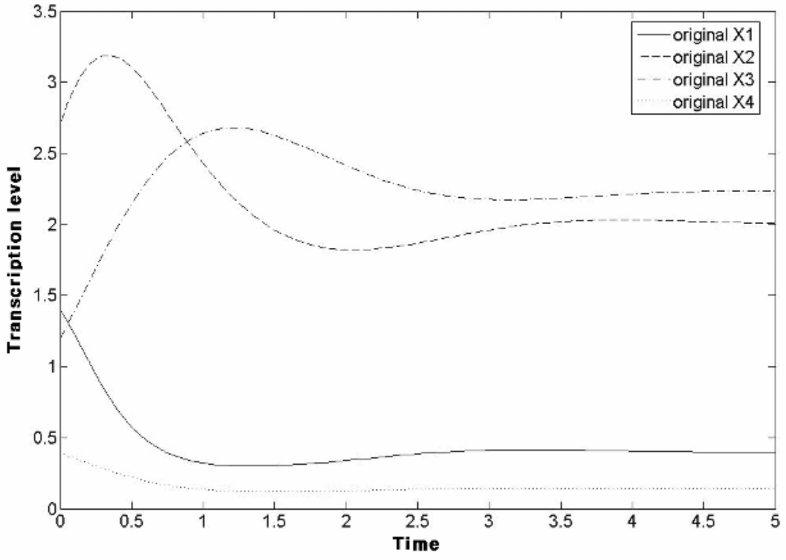
3 Experimental Results

3.1 Data

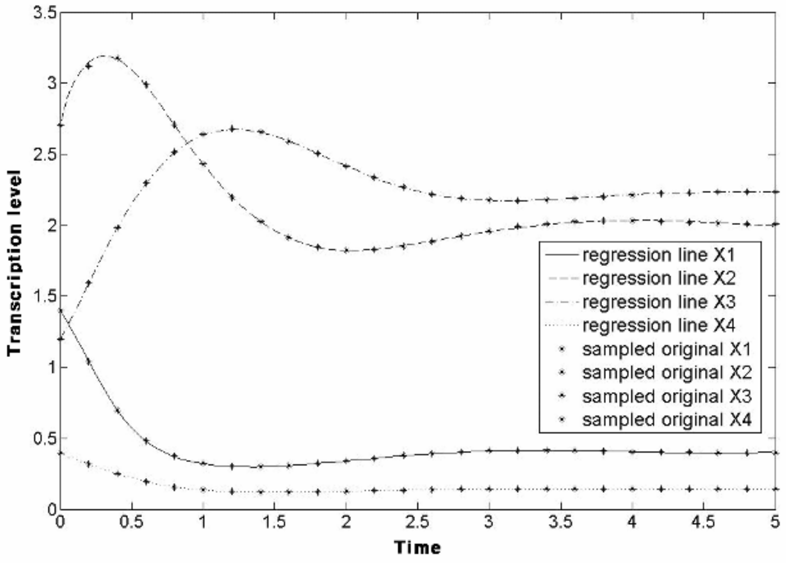
To evaluate the performance of the proposed algorithms, we consider the artificial gene regulatory network which came from [13]. This gene regulatory network modeled 4 reactants influenced with one another and all reactants are auto-regulated. The S-system model for the used gene regulatory network is represented in Fig. 2(a) and (b) and the time-course graph of the given gene regulatory network is represented in Fig. 4(a). To create artificial time-course profiles, we solve this true S-system with the initial values, $X_1(0)=1.4$, $X_2(0)=2.7$, $X_3(0)=1.2$, and $X_4(0)=0.4$ by using the 4th-order Runge-Kutta method. The size of sampled data point is 25 and their time intervals are 0.2 seconds as shown in Fig. 4(b). We set 4 times smaller number of time points than that of the previous study [13] since it is very difficult to obtain a lot of time-course data from the real biological measurements.

3.2 Results of the Genetic Programming

We use Frayn's GPLib library [18] for the symbolic regression with GP. To evolve the mathematical models for the given data, we use the function set $F = \{+, -, \times, /, \wedge, \sin, \exp, \log, \text{sqrt}\}$ and terminal set $T = \{t, \Re, \pi\}$, where $\Re$ is real constant and t is the time point. The population size is 3,000 and the maximum number of generations is 2,000. Tournament selection is used and its size is 4. Crossover rate is 0.35 and mutation rate is 0.5. We set the length penalty of the genetic program as zero for the accurate curve fitting. We generate 100 points and derivations from the obtained models for the input of the next stage, that is, the hybrid evolutionary algorithms.



(a) original graph



(b) sample data points and regression results by GP

Fig. 4. The true (a) simulated (b) time-series data for the artificial genetic network

Results of the genetic programming step are as follows:

f_1(t)=(sqrt((((sqrt(((sqrt(2.957153))-((sin(sqrt(t)))+((sin((1.854912)-(((sqrt((3)*(t)))
(sqrt(t)))+(4.435898))))+(-2.355442))))(t)))+((40.675830)/(exp((t)*((sqrt(t))-
((sin(((sin((((sqrt((3.756391)*(t)))*(sqrt(t)))+(log(86)))+(7)))+(3))+(sin(sin((1.654737
)*(t))))))+(-2.355442)))))))/(sqrt(54.598150)))/(sqrt(7.931547)))),

f_2(t)=(sqrt(((3.777992)-((((((4.190957)-((t)-((sin((((t)*((t)^(t)))-((((t)*((2.883554)
/((4)-(log(t)))))+(2.791190))-((exp((t)-(2.226704)))^(sqrt(9.642561)))))/((t)^(t))))
*(2.678347))))/(3.462360))+(3.792098))-(4.796861))/(4)))+(((((3.792098)-((exp(t))
/(3.462360)))-((t)*((t)^(t)))) /((t)^(t)))/((t)^(t))))),

f_3(t)=((log(log(exp(log(log(exp(exp(((log(exp(π)))-((sin((9)+(((t)+(8.000000))
^(sqrt(1.420245)))))/(((exp(t))*(379.000000))/(84.000000))))-((sin((8)+(((t)
+(((log(109))-(1.258803))/(6.620476)))^(sqrt(log(4.059461))))))
/(((exp(((8.337047)*((log(log(sqrt(3.021610))))+(2.000000)))-(5.912041)))*(exp(t)))
/(85.000000))))))))))^(5.933873)),

f_4(t)=((((log(6.249382))^((sqrt(6))*(((sqrt(10.000000))^((1.172486)-(t)))
/(6.981835))))^((1.161464)-((1.161464)/(((((sqrt(6.249382))*((log(7.008566))
*((((((exp((6.980522)/((sqrt(6.288201))^(1.344547))))^((1.735082)-(t)))
/(0.290257))*(sqrt(6.000000)))^(((9.704760)^((-0.050358)-(t)))-(t)))/(0))))
^((1.634223)+((7.277223)^((0.290257)-(t)))))^(0.161464))/(t)))))/(6.980522)).

Fig. 4 shows the true profiles and estimated curves and sampled data points from the genetic programming and we confirm that the GP recover the true dynamics of the S-system

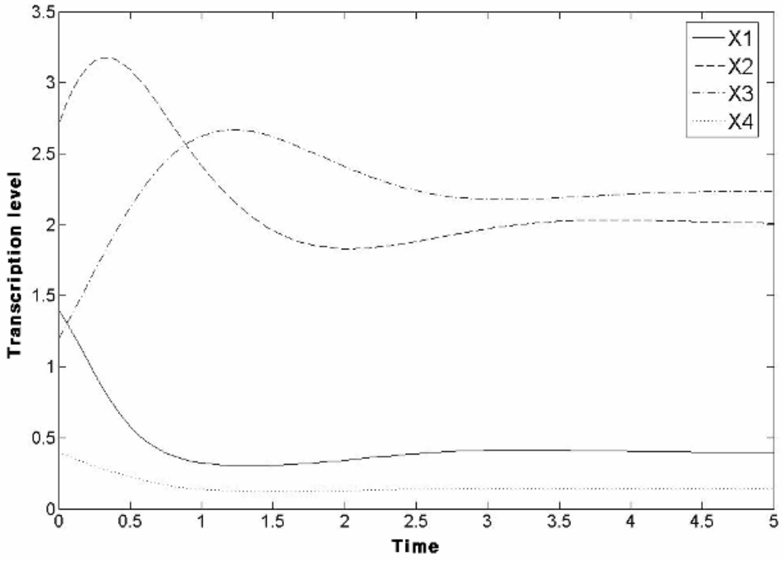
3.3 Results of the Hybrid Evolutionary Algorithms

Using 100 points obtained from the GP step, we search the S-system by the hybrid evolutionary algorithms. This step is composed with steady-state evolutionary algorithms with local optimization procedure - (1+1) evolutionary strategy. For the sparse network structures, we adopt a structural constraint which the evolved networks should satisfy. That is, each gene is assumed to be related to one or two other genes in the system. Hence, the randomly generated initial individuals and offspring by crossover and mutation operators should have one or two non-zeros elements in g and h. Crossover rate is 0.8 and mutation rate is 0.3. As a local optimization for the parameter values, (1+1) evolutionary strategy is performed for 80 fitness evaluations. The search ranges of the parameters are [0.0, 15.0] for α_i and β_i, and [-1.0, 1.0] for g_{ij} and h_{ij}. With the population size of 10^4, the proposed algorithm successfully identified the true structure after 10^6 generation. As shown in Fig. 5, we can also recover the dynamic profiles with the estimated parameters.

$$\alpha = \begin{pmatrix} 14.614 \\ 8.123 \\ 3.088 \\ 2.866 \end{pmatrix} \qquad g = \begin{pmatrix} 0.0 & 0.0 & -0.623 & 0.0 \\ 0.486 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.761 & 0.0 & 0.0 \\ 0.254 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

$$\beta=\begin{pmatrix}12.670\\3.076\\5.583\\5.391\end{pmatrix}\qquad h=\begin{pmatrix}0.398 & 0.0 & 0.0 & 0.0\\0.0 & 0.753 & 0.0 & 0.0\\0.0 & 0.0 & 0.486 & 0.231\\0.0 & 0.0 & 0.0 & 0.443\end{pmatrix}$$

(a) the S-system obtained by our hybrid EAs



(b) time course profiles of the identified system

Fig. 5. The results of the proposed hybrid evolutionary algorithms

4 Conclusion

We propose multi-stage evolutionary algorithms to identify gene regulatory networks efficiently with the S-system representation. We adopt the pre-processing symbolic regression step by genetic programming for avoiding the time-consuming numerical integration. We also develop hybrid genetic algorithms and modify the fitness function to identify the structure of gene regulatory networks and to estimate the corresponding parameters simultaneously without the threshold values for the sparse network structures. By applying the proposed method to the identification of an artificial genetic network, we verify its capability of finding the true S-system.

One important future work is to demonstrate the usefulness of the proposed algorithm for real experimental biological data such as the gene expression profiles from the microarrays and NMR measurements of some metabolites. As the by-product of the population diversity maintenance of our evolutionary algorithms, we will be able to attain the different plausible topologies for the network very efficiently. These can be reliable candidates to the biologists who want to discover unknown interactions among some components in the genetic networks.

Acknowledgement

This work was supported by the Korea Ministry of Science and Technology through National Research Lab (NRL) project and the Ministry of Education and Human Resources Development under the BK21-IT Program. The ICT at the Seoul National University provided research facilities for this study.

References

1. Covert, M.W., Schilling, C.H., Famili, I. Edwards, J.S., Goryanin, I.I. Selkov, E., Palsson, B.O.: Metabolic modeling of microbial strains in silico. *Trends in Biochemical Science* 26 (2001) 179-186
2. De Jong, H.: Modeling and simulation of genetic regulatory system: a literature review. *Journal of Computational Biology* 9 (2002) 67-103
3. Stelling, J.: Mathematical models in microbial systems biology. *Current Opinion in Microbiology* 7 (2004) 513-518
4. Barabasi, A.-L., Oltvai, Z.N.: Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics* 5 (2004) 101-113
5. De Jong, H., Gouze, J.-L., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathematical Biology* 66 (2004) 301-340
6. Rao, C.V., Wolf, D.M., Arkin, A.P.: Control, exploitation and tolerance of intracellular noise. *Nature* 402 (2002) 231-237
7. Voit, E.O.: *Computational Analysis of Biochemical Systems.* Cambridge University Press (2000)
8. Fogel, D.B.: *System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling.* Ginn Press (1991)
9. Tominaga, D., Koga, N., Okamoto, M.: Efficient numerical optimization algorithm based on genetic algorithm for inverse problem. In: Whitley, D. et al. (Eds.), *Proceedings of the 2000 Genetic and Evolutionary Computation Conference* (2000) 251-258
10. Ando, S., Sakamoto, E., Iba, H.: Evolutionary modeling and inference of gene network. *Information Science* 145 (2002) 237-259
11. Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., Tomita, M.: Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics* 19 (2003) 643-650
12. Spieth, C., Streichert, F., Speer, N., Zell, A.: Optimizing topology and parameters of gene regulatory networt models from time-series experiments. In: Deb, K. et al. (eds.): *Proceedings of the 2004 Genetic and Evolutionary Computation Conference.* Lecture Notes in Computer Science, Vol. 3102. Springer-Verlag (2004) 461-470
13. Almeida J.S., Voit E.O.: Neural network-based parameter estimation in S-system models of biological networks, *Genome Informatics* 14 (2003) 114-123
14. Tsai, K.-Y., Wang, F.-S.: Evolutionary optimization with data collocation for reverse engineering of biological networks. *Bioinformatics* 21 (2005) 1180-1188
15. Savageau, M.A.: *Biochmical System Analysis: A Study of Function and Design in Molecular Biology,* Addison-Wesley (1976)

16. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press (1992)
17. Bäck, T.: *Evolutionary Algorithm in Theory and Practice,* Oxford University Press (1996)
18. http://www.cs.bham.ac.uk/~cmf/GPLib/GPLib.html
19. Harik, G.R.: Finding multimodal solutions using restricted tournament selection. In: Eshelman, L.J. (ed.): *Proceedings of the Sixth International Conference on Genetic Algorithms* (1995) 24-31

Human Papillomavirus Risk Type Classification from Protein Sequences Using Support Vector Machines

Sun Kim and Byoung-Tak Zhang

Biointelligence Laboratory,
School of Computer Science and Engineering,
Seoul National University, Seoul 151-744, South Korea
{skim, btzhang}@bi.snu.ac.kr

Abstract. Infection by the human papillomavirus (HPV) is associated with the development of cervical cancer. HPV can be classified to high- and low-risk type according to its malignant potential, and detection of the risk type is important to understand the mechanisms and diagnose potential patients. In this paper, we classify the HPV protein sequences by support vector machines. A string kernel is introduced to discriminate HPV protein sequences. The kernel emphasizes amino acids pairs with a distance. In the experiments, our approach is compared with previous methods in accuracy and F1-score, and it has showed better performance. Also, the prediction results for unknown HPV types are presented.

1 Introduction

The cervical cancer is a leading cause of cancer death among women worldwide. Epidemiologic studies have shown that the association of genital human papillomavirus (HPV) with cervical cancer is strong, independent of other risk factors [1]. HPV infection causes virtually all cases of cervical cancer because certain high-risk HPVs develop cancer even though most cases of HPV are low-risk and rarely develop into cancer. Especially, high-risk HPV types could induce more than 95% of cervical cancer in woman.

The HPV is a relatively small, double-strand DNA tumor virus that belongs to the papovavirus family (papilloma, polyoma, and simian vacuolating viruses). More than 100 human types are specific for epithelial cells including skin, respiratory mucosa, or the genital tract. And the genital tract HPV types are classified into two or three types such as low-, intermediate-, and high-risk types by their relative malignant potential [2]. The common, unifying oncogenic feature of the vast majority of cervical cancers is the presence of HPV, especially high-risk type HPV [3]. Thus the risk type detection of HPVs have become one of the most essential procedures in cervical cancer remedy. Currently, the HPV risk types are still manually classified by experts, and there is no deterministic method to expect the risk type for unknown or new HPVs.

Since the HPV classification is important in medical judgments, there have been many epidemiological and experimental studies to identify HPV risk types

[3]. Polymerase chain reaction (PCR) is a sensitive technique for the detection of very small amounts of HPV's nucleic acids in clinical specimens. It has been used in most epidemiological studies that have evaluated the role of these viruses in cervical cancer causation [4]. Bosch et al. [1] investigated epidemiological characteristic that whether the association between HPV infection and cervical cancer is consistent worldwide in the context of geographic variation. Burk et al. [5] inspected the risk factors for HPV infection in 604 young college women through examining social relationship and detected various factors of HPV infection with L1 consensus primer PCR and Southern blot hybridization. Muñoz et al. [6] classified the HPV risk types with epidemiological experiments based on risk factor analysis. They pooled real data from 1,918 cervical cancer patients and analyzed it by PCR based assays.

Detection of HPV risk types can be a protein function prediction even though functions are described at many levels, ranging from biochemical function to biological processes and pathways, all the way up to the organ or organism level [7]. Many approaches for protein function prediction are based on similarity search between proteins with known function. The similarity among proteins can be defined in a multitude of ways [8]: sequence alignment, structure match by common surface clefts or binding sites, common chemical features, or certain motifs comparison. However, none of the existing prediction systems can guarantee generally good performance. Thus it is required to develop classification methods for HPV risk types. Eom et al. [9] presented a sequence comparison method for HPV classification. They use DNA sequences to discriminate risk types based on genetic algorithms. Joung et al. [10] combined with several methods for the risk type prediction from protein sequences. Protein sequences are first aligned, and the subsequences in high-risk HPVs against low-risk HPVs are selected by hidden Markov models. Then a support vector machine is used to determine the risk types. The main drawback of this paper is that the method is biased by one sequence pattern. Alternatively, biomedical literature can be used to predict HPV risk types [11]. But, text mining approaches have the limitation for prediction capability because they only depend on texts to capture the classification evidence, and the obvious keywords such as 'high' tend to be appeared in the literature explicitly.

In this paper, we propose a method to classify HPV risk types using protein sequences. Our approach is based on support vector machines (SVM) to discriminate low- and high-risk types and a string kernel is introduced to deal with protein sequences. The string kernel first maps to the space consisting of all subsequences of amino acids pair. A RBF kernel is then used for nonlinear mapping into a higher dimensional space and similarity calculation. Especially, the proposed kernel only uses amino acids of both ends in k-length subsequences to improve the classification performance. It is motivated by the assumption that amino acids pairs with certain distance affects the HPV's biological function, i.e. risk type, more than consecutive amino acids. The experimental results show that our approach provides better performance than previous approaches in accuracy and F1-score.

Our work addresses how to classify HPV risk types from protein sequences by SVM approaches, which can provide a guide to determine unknown or new HPVs. The paper is organized as follows. In Section 2, we explain the SVM method for classification. Then the string kernel for HPV protein sequence is presented in Section 3. In Section 4, we present the experimental results and draw conclusions in Section 5.

2 Support Vector Machine Classifiers

We use support vector machines to classify HPV risk types. A string kernel-based SVM is trained on HPV protein sequences and tested on unknown sequences. Support vector machines have been developed by Vapnik to give robust performance for classification and regression problems in noisy, complex data [12]. It has been widely used from text categorization to bioinformatics in recent days. When it is used for classification problem, a kernel and a set of labeled vectors, which is marked to positive or negative class are given. The kernel functions introduce nonlinear features in hypothesis space without explicitly requiring nonlinear algorithms. SVMs learn a linear decision boundary in the feature space mapped by the kernel in order to separate the data into two classes.

For a feature mapping ϕ, the training data $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, is mapped into the feature space $\Phi(S) = \{\Phi(\mathbf{x}_i), y_i\}_{i=1}^n$. In the feature space, SVMs learn the hyperplane $f = \langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b, \mathbf{w} \in \mathbb{R}^N, b \in R$, and the decision is made by $sgn(\langle \mathbf{w}, \Phi(\mathbf{x})\rangle + b)$. The decision boundary is the hyperplane $f = 0$ and its margin is $1/\|\mathbf{w}\|$. SVMs find a hyperplane that has the maximal margin from each class among normalized hyperplanes.

To find the optimal hyperplane, it can be formulated as the following problem:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{1}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b) \geq 1, \; i = 1, \ldots, n. \tag{2}$$

By introducing Lagrange multipliers $\alpha_i \geq 0, \; i = 1, \ldots, n$, we get the following dual optimization problem:

$$\text{maximize} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle \tag{3}$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, n, \tag{4}$$

$$\sum_{i=1}^n \alpha_i y_i = 0. \tag{5}$$

By solving this dual problem, one obtains optimal solution $\alpha_i, 1 \leq i \leq n$, which needs to determine the parameters $(\mathbf{w}, b)$. For the solution $\alpha_i, \ldots, \alpha_n$, the nonlinear decision function $f(\mathbf{x})$ is expressed in terms of the following parameters:

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}_i)\rangle + b\right) \quad (6)$$

$$= \mathrm{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right). \quad (7)$$

We can work on the feature space by using kernel functions, and any kernel function K satisfying Mercer's condition can be used.

3 Kernel Function

For HPV protein classification, we introduce a string kernel based on the spectrum kernel method. The spectrum kernel was used to detect remote homology detection [13][14]. The input space $\mathcal{X}$ consists of all finite length sequences of characters from an alphabet $\mathcal{A}$ of size $|\mathcal{A}| = l$ ($l = 20$ for amino acids). Given a number $k \geq 1$, the k-spectrum of a protein sequence is the set of all possible k-length subsequences (k-mers) that it contains. The feature map is indexed by all possible subsequences a of length k from $\mathcal{A}$. The k-spectrum feature map $\Phi_k(x)$ from $\mathcal{X}$ to $\mathbb{R}^{l^k}$ can be defined as:

$$\Phi_k(x) = (\phi_a(x))_{a \in \mathcal{A}^k}. \quad (8)$$

where $\phi_a(x)$ = number of occurrences of a occurs in x. Thus the k-spectrum kernel function $K^s(x_i, x_j)$ for two sequences x_i and x_j is obtained by taking the inner product in feature space:

$$K_k^s(x_i, x_j) = \langle \Phi_k(x_i), \Phi_k(x_j)\rangle. \quad (9)$$

To fit in with HPV risk type classification, we want to modify the spectrum kernel. Proteins are linear chains of amino acids, which are made during the process of translation, and it is called primary structure. The natural shape of proteins are not such as straight lines, rather 3-dimensional structures formed by protein folding, which is a consequence of the primary structure. The structure of a similar homologous sequence can be helpful to identify the tertiary structure of the given sequence. Here, we assume that the amino acids pair with certain distance affect HPV's risk type function more than consecutive amino acids according to its 3-dimensional structure property, and the HPV risk types can be identified by the amino acids pair with a fixed distance, which mostly influence on risk type decision. This assumption is somewhat rough, but it can be useful for relatively short and α helix-dominant sequences.

Under the assumption, we want to define a string kernel, the gap-spectrum kernel based on k-spectrum. For a fixed k-mer $a = a_1 a_2 \ldots a_k$, $a_i \in \mathcal{A}$, 2-length sequence $\beta = a_1 a_k$, $\beta \in \mathcal{A}^2$. β indicates the amino acids pair with (k-2) gap. The feature map $\Psi_k(x)$ is defined as:

Table 1. The manually classified risk types of 72 HPVs

Type	Class	Type	Class	Type	Class	Type	Class
HPV1	Low	HPV20	Low	HPV38	Low	HPV57	?
HPV2	Low	HPV21	Low	HPV39	High	HPV58	High
HPV3	Low	HPV22	Low	HPV40	Low	HPV59	High
HPV4	Low	HPV23	Low	HPV41	Low	HPV60	Low
HPV5	Low	HPV24	Low	HPV42	Low	HPV61	High
HPV6	Low	HPV25	Low	HPV43	Low	HPV63	Low
HPV7	Low	HPV26	?	HPV44	Low	HPV65	Low
HPV8	Low	HPV27	Low	HPV45	High	HPV66	High
HPV9	Low	HPV28	Low	HPV47	Low	HPV67	High
HPV10	Low	HPV29	Low	HPV48	Low	HPV68	High
HPV11	Low	HPV30	Low	HPV49	Low	HPV70	?
HPV12	Low	HPV31	High	HPV50	Low	HPV72	High
HPV13	Low	HPV32	Low	HPV51	High	HPV73	Low
HPV15	Low	HPV33	High	HPV52	High	HPV74	Low
HPV16	High	HPV34	Low	HPV53	Low	HPV75	Low
HPV17	Low	HPV35	High	HPV54	?	HPV76	Low
HPV18	High	HPV36	Low	HPV55	Low	HPV77	Low
HPV19	Low	HPV37	Low	HPV56	High	HPV80	Low

$$\Psi_k(x) = (\phi_\beta(x))_{\beta \in \mathcal{A}^2} . \tag{10}$$

where $\phi_\beta(x)$ = number of occurrences of β occurs in x. Furthermore a nonlinear kernel function, RBF kernel is appended to increase the discrimination ability between HPV risk types. By closure properties of kernels [15], the gap-spectrum kernel is defined as follows:

$$K_k(x_i, x_j) = K'(\Psi_k(x_i), \Psi_k(x_j)) \tag{11}$$

$$= \exp\left(-\gamma \|\Psi_k(x_i) - \Psi_k(x_j)\|^2\right) . \tag{12}$$

where $\gamma > 0$. This string kernel is used in combination with the SVM explained in Section 2.

4 Experimental Results

4.1 Data Set

In this paper, we use the HPV sequence database in Los Alamos National Laboratory (LANL) [16], and total 72 types of HPV are used for experiments. The risk types of HPVs were determined based on the HPV compendium (1997). If a HPV belongs to skin-related or cutaneous groups, the HPV is classified into low-risk type. On the other hand, a HPV is classified as a high-risk if it is known to be high-risk type for cervical cancer. The comments in LANL database are used to decide risk types for some HPVs, which are difficult to be classified. Seventeen sequences out of 72 HPVs were classified as high-risk types (16, 18,

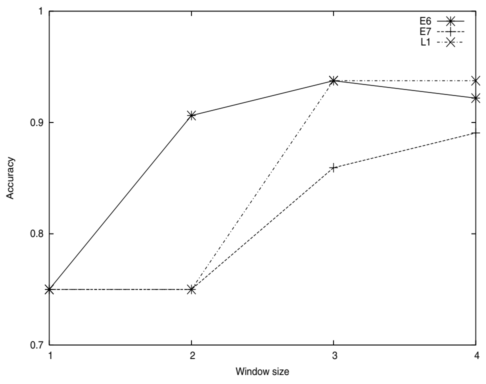


Fig. 1. SVM classification performance by window size

31, 33, 35, 39, 45, 51, 52, 56, 58, 59, 61, 66, 67, 68, and 72), and others were classified as low-risk types. Table 1 shows the HPV types and their classified risk type. The symbol '?' in the table denotes unknown risk type that cannot be determined.

Since several proteins can be applied to discriminate HPVs, we have evaluated the classification accuracy using the SVM with RBF kernel to determine the gene products to be used for the experiments. The input data is the normalized frequency vector by sliding window method. It has been performed to decide the most informative protein among gene products for HPV risk type classification. Figure 1 depicts the accuracy changes by window size for E6, E7, and L1 proteins. The accuracy is the result of leave-one-out cross-validation. It indicates that the accuracy using E6 protein is mostly higher than using E7 and L1 proteins. However, the overall accuracy gets high by increasing window size for all proteins because the HPV sequences are relatively short and unique patterns are more generated when window size is long. That is, the learners overfit protein sequences for long window size. Viral early proteins E6 and E7 are known for inducing immortalization and transformation in rodent and human cell types. E6 proteins produced by the high-risk HPV types can bind to and inactivate the tumor suppressor protein, thus facilitating tumor progression [16][17]. This process plays an important role in the development of cervical cancer. For these reasons, we have chosen E6 protein sequences corresponding to the 72 HPVs.

4.2 Evaluation Measure

For the HPV prediction, it is important to get high-risk HPVs as many as possible, although a few low-risk HPVs are misclassified, hence we evaluate the system performance using F1-score rather than Matthews correlation coefficient. F1-score is a performance measure usually used for information retrieval systems, and it is effective to evaluate how well the classifier did when it assigned classes such as high-risk type.

Table 2. The contingency table to evaluate the classification performance

		Risk type answer	
		High	*Low*
Prediction result	*High*	*a*	*b*
	Low	*c*	*d*

When binary scales are used for both answer and prediction, a contingency table is established showing how the data set is divided by these two measures (Table 2). By the table, the classification performance is measured as follows:

$$\begin{aligned} precision &= \frac{a}{a+b} \cdot 100\% \\ recall &= \frac{a}{a+c} \cdot 100\% \\ F1 - score &= \frac{2 \cdot precision \cdot recall}{precision + recall}. \end{aligned}$$

4.3 HPV Classification

We have tested the gap-spectrum SVM method using E6 protein sequences. Leave-one-out cross-validation is used to determine the classification performance. Figure 2 shows the accuracy changes according to k given in Equation (12). The graph shows the accuracy has the highest performance (97.22%) when $k = 4$, and the performance is decreases as it gets more or less k. $k = 4$ means that we only use the amino acids pairs which have two gaps between them for HPV classification. $k = 2$ is exactly same as the SVM using RBF kernel with 2-spectrum method, and the accuracy is 94.44% for $k = 2$. Even though it gives higher score than other methods as shown in Figure 3, the kernel methods with $k > 2$ still gives better performance. As a result, the amino acids pair with a distance can provide more evidence than consecutive amino acids to discriminate low- and high-risk HPV proteins.

The final classification performance in accuracy and F1-score is given in Figure 3. It compares with previous results using SVM approaches based on sequence alignment and text mining approaches. The SVM method which utilizes alignment information has been reported in [10]. AdaCost and naïve Bayes are text mining methods using HPV literature data, which have been reported in [11]. Our approach shows 97.22% of accuracy and 95.00% of F1-score, while previous SVM method shows 93.15% of accuracy and 85.71% of F1-score. For text-based classification, the AdaCost method shows 93.05% of accuracy and 86.49% of F1-score, and the naïve Bayes method shows 81.94% of accuracy and 63.64% of F1-score. Additionally, the accuracy obtained from the DNA sequence-based method [9] is 85.64%. It is interesting that it gets relatively higher score in F1-score than in accuracy. F1-score is related with the number of high-risk HPVs found by classifiers, while accuracy is related with the number of HPVs which is correctly classified. Therefore, F1-score is more important than accuracy in this task.

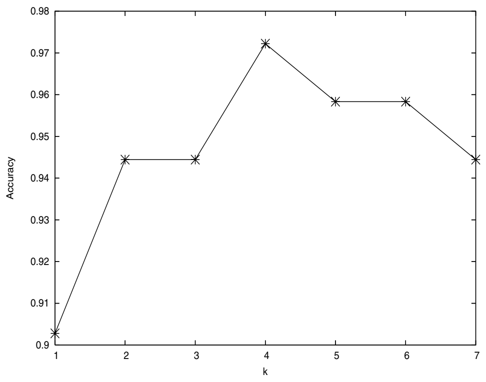


Fig. 2. Accuracy changes by the gap-spectrum kernel

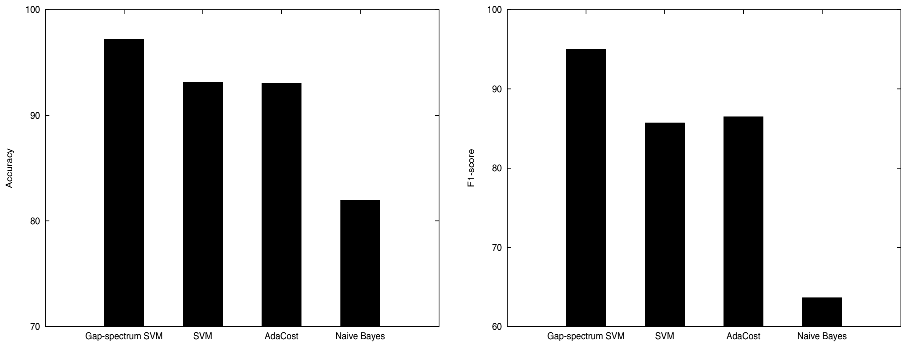


Fig. 3. The performance comparison of proposed approaches and previous classification methods

Text mining approaches only depend on the clues from text sentences. If the text documents are unavailable for unknown HPVs, there is no way to classify them, whereas the sequence-based classification does not need to use any additional information except sequence itself.

Table 3 shows the risk type prediction for HPVs marked as unknown in Table 1. HPV26, HPV54, HPV57, and HPV70 are predicted as high-, low-, low-, and high-risk, respectively. The prediction results for HPV26 and HPV54 are identical to the one in Muñoz et al. [6], and we assume that their results are correct because it is based on epidemiologic classification from over 1,900 patients. For HPV70, there are different decisions for the risk type according to previous research [6][18][19], and the risk type of HPV57 cannot be decided yet because of insufficient previous works. By the prediction results, we can conclude our approach provides certain probability for whether unknown HPVs are high-risk or not.

Table 3. Predicted risk type for unknown HPVs

Type	HPV26	HPV54	HPV57	HPV70
Risk	High	Low	Low	High

5 Conclusion

We have presented a machine learning approach to classify HPV risk types. Our method uses the SVM classifier with the gap-spectrum kernel based on k-spectrum methods. The proposed kernel is designed to emphasize amino acids pair with a fixed distance, which can be suitable for relatively short and α helix-dominant sequences. For the experiments, the performance has been measured based on leave-one-out cross-validation. According to experimental results, amino acids pair with a fixed distance provides good performance to discriminate HPV proteins by its risk. Especially, it is important not to have false negatives as many as possible in this task. Therefore F1-score is important because it considers both precision and recall based on high-risk type. Our approach shows significant improvement in F1-score as compared with previous methods, and the prediction for unknown HPV types has given promising results. We can conclude that the relationship between amino acids with $k = 4$ supports important role to divide low- and high-risk function in HPV E6 proteins.

In this paper, we consider all protein subsequences equally. Even though SVMs naturally detect the important factors in a high-dimensional space, it is necessary to analyze what components are more informative for HPV risk types. Also, protein structure or biological literature information can be combined with this method for more accurate prediction. Thus, study on exploring efficient analysis method remains as future works.

Acknowledgement

This work was supported by the Korea Ministry of Science and Technology through National Research Lab (NRL) project and the Ministry of Education and Human Resources Development under the BK21-IT Program. The ICT at the Seoul National University provided research facilities for this study.

References

[1] Bosch, F. X., Manos, M. M., et al.: Prevalence of Human Papillomavirus in Cervical Cancer: a Worldwide Perspective. *Journal of the National Cancer Institute* **87** (1995) 796–802.

[2] Janicek, M. F. and Averette, H. E.: Cervical Cancer: Prevention, Diagnosis, and Therapeutics. *Cancer Journals for Clinicians* **51** (2001) 92–114.

[3] Furumoto, H. and Irahara, M.: Human Papillomavirus (HPV) and Cervical Cancer. *Journal of Medical Investigation* **49** (2002) 124–133.

[4] Centurioni, M. G., Puppo, A., et al.: Prevalence of Human Papillomavirus Cervical Infection in an Italian Asymptomatic Population. *BMC Infectious Diseases* **5**(77) (2005).

[5] Burk, R. D., Ho, G. Y., et al.: Sexual Behavior and Partner Characteristics Are the Predominant Risk Factors for Genital Human Papillomavirus Infection in Young Women. *The Journal of Infectious Diseases* **174** (1996) 679–689.

[6] Muñoz, N., Bosch, F.X., et al.: Epidemiologic Classification of Human Papillomavirus Types Associated with Cervical Cancer. *New England Journal of Medicine* **348** (2003) 518–527.

[7] Watson, J. D., Laskowski, R. A., and Thornton, J. M.: Predicting Protein Function from Sequence and Structural Data. *Current Opinion in Structural Biology* **15** (2005) 275–284.

[8] Borgwardt, K. M., Ong, C. S., et al.: Protein Function Prediction via Graph Kernels. In *Proceedings of Thirteenth International Conference on Intelligenc Systems for Molecular Biology* (2005) 47–56.

[9] Eom, J.-H., Park, S.-B., and Zhang, B.-T.: Genetic Mining of DNA Sequence Structures for Effective Classification of the Risk Types of Human Papillomavirus (HPV). In *Proceedings of the 11th International Conference on Neural Information Processing* (2004) 1334–1343.

[10] Joung, J.-G., O, S.-J, and Zhang, B.-T.: Prediction of the Risk Types of Human Papillomaviruses by Support Vector Machines. In *Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence* (2004) 723–731.

[11] Park, S.-B., Hwang, S., and Zhang, B.-T.: Mining the Risk Types of Human Papillomavirus (HPV) by AdaCost. In *Proceedings of the 14th International Conference on Database and Expert Systems Applications* (2003) 403–412.

[12] Vapnik, V. N.: Statistical Learning Theory. Springer (1998).

[13] Leslie, C., Eskin, E., and Noble, W. S.: The Spectrum Kernel: A String Kernel for SVM Protein Classification. In *Proceedings of the Pacific Symposium on Biocomputing* (2002) 564–575.

[14] Leslie, C., Eskin, E., et al.: Mismatch String Kernels for Discriminative Protein Classification. *Bioinformatics* **20**(4) (2004) 467–476.

[15] Shawe-Taylor, J. and Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004).

[16] The HPV sequence database in Los Alamos laboratory, http://hpv-web.lanl.gov/stdgen/virus/hpv/index.html.

[17] Pillai, M., Lakshmi, S., et al.: High-Risk Human Papillomavirus Infection and E6 Protein Expression in Lesions of the Uterine Cervix. *Pathobiology* **66** (1998) 240–246.

[18] Longuet, M., Beaudenon, S., and Orth, G.: Two Novel Genital Human Papillomavirus (HPV) Types, HPV68 and HPV70, Related to the Potentially Oncogenic HPV39. *Journal of Clinical Microbiology* **34** (1996) 738–744.

[19] Meyer, T., Arndt, R., et al.: Association of Rare Human Papillomavirus Types with Genital Premalignant and Malignant Lesions. *The Journal of Infectious Diseases* **178** (1998) 252–255.

Hierarchical Clustering, Languages and Cancer

Pritha Mahata[1,2], Wagner Costa[1], Carlos Cotta[3], and Pablo Moscato[1,2]

[1] Newcastle Bioinformatics Initiative,
School of Electrical Engineering and Computer Science,
The University of Newcastle, Callaghan, NSW, 2308, Australia
[2] Australian Research Centre in Bioinformatics
[3] Dept. Lenguajes y Ciencias de la Computación, University of Málaga,
ETSI Informática, Campus de Teatinos, Málaga – 29071, Spain
Pablo.Moscato@newcastle.edu.au

Abstract. In this paper, we introduce a novel objective function for the hierarchical clustering of data from distance matrices, a very relevant task in Bioinformatics. To test the robustness of the method, we test it in two areas: (a) the problem of deriving a phylogeny of languages and (b) subtype cancer classification from microarray data. For comparison purposes, we also consider both the use of *ultrametric trees* (generated via a two-phase evolutionary approach that creates a large number of hypothesis trees, and then takes a consensus), and the best-known results from the literature.

We used a dataset of measured 'separation time' among 84 Indo-European languages. The hierarchy we produce agrees very well with existing data about these languages across a wide range of levels, and it helps to clarify and raise new hypothesis about the evolution of these languages.

Our method also generated a classification tree for the different cancers in the NCI60 microarray dataset (comprising gene expression data for 60 cancer cell lines). In this case, the method seems to support the current belief about the heterogeneous nature of the ovarian, breast and non-small-lung cancer, as opposed to the relative homogeneity of other types of cancer. However, our method reveals a close relationship of the melanoma and CNS cell-lines. This is in correspondence with the fact that metastatic melanoma first appears in central nervous system (CNS).

1 Introduction

A large number of articles in bioinformatics use single-objective unsupervised hierarchical clustering algorithms to identify "subtypes". Biologically-oriented journals, in general, have low requirements in terms of algorithmic reproducibility. The validation of the algorithms used in different problem scenarios is either poor or non-existing. Working on a dataset of measured separation times between 84 Indo-European languages we started to notice the deficiencies of a large number of hierarchical clustering schemes. The same problems occur when analyzing datasets derived from mitochondrial DNA distances among species [1].

In this paper, we pose the clustering problem as a graph optimization problem and propose a novel objective function which performs very well in diverse types of datasets. We start with a distance matrix for a set of objects and compute a weighted graph in which vertices represent objects and edges are weighted by the distance between the corresponding vertices. Then our objective function tries to obtain a solution whose fitness is maximal and proportional to the sum of the weights on the edges between two sets of vertices, and to the sum of the reciprocals of the weights on the edges inside the sets. We denote this as *arithmetic-harmonic cut*. The recursive application of such cuts generates a tree-based classification of the data. While our primary concern is the classification of microarray data, we are also interested in testing the robustness of the approach, validating it in other domains. For this purpose, we show results for two different datasets: (a) a dataset for 84 Indo-European languages, and (b) a dataset for 60 cancerous cell-lines (NCI60). Next section will provide more details on the algorithmic methods we have used.

2 Hierarchical Clustering Methods Considered

In addition to the clustering solutions available in the literature for the datasets considered, we have used two unsupervised techniques for computing alternative solutions. The first one is based on arithmetic-harmonic cuts, and the second one relies on the utilization of ultrametric trees. These will be described below.

2.1 Arithmetic-Harmonic Cuts

The method of arithmetic-harmonic cuts approaches the construction of the hierarchy in a top-down fashion. To be precise, it can be described as a recursive process in which we solve a graph optimization problem at each step. Let $G(E, V, W)$ be an undirected, complete weighted graph with no self-loops and such that the weight of any edge is a positive integer number (i.e., $w(e) > 0$) representing distance or some measure of dissimilarity between a pair of objects. We first find a partition of the set V of vertices into $\{S, V \setminus S\}$, which generates a partition of the set E of edges in two sets E_{in} and E_{out}. The set $E_{out} \subset E$ is the set of edges that link a vertex in S and a vertex in $V \setminus S$ (similarly, $E_{in} = E \setminus E_{out}$ is the set of edges connecting vertices in the same partition). Such a partition is defined by maximizing the following objective function

$$F = \left(\sum_{e \in E_{out}} w(e) \right) \left(\sum_{e \in E_{in}} 1/w(e) \right) \tag{1}$$

We have implemented an exact backtracking algorithm and also a memetic algorithm (similar to the work of Merz and Freisleben [2] for Graph Bipartitioning) as a meta-heuristic to calculate the best partitioning of the vertices for a given graph. The difference with respect to [2] is that we remove the constraint of equal partitioning of the graph in our memetic algorithm. Thus, the memetic algorithm uses (a) a differential greedy algorithm (similar to that in [3]) for initialization of a set of solutions for the problem, (b) a differential greedy crossover (a modification of the algorithm in [2]) for evolution of the population, and (c) a variable neighborhood local search (see [4]) to improve the newly generated solutions. Whenever the population stagnates, we keep the best solution and re-initialize the rest of solutions in the set. We use this memetic algorithm if the graph contains more than 25 vertices, and a backtracking enumeration algorithm otherwise. Notice that even though backtracking gives us an optimal solution, a memetic algorithm may not. However, in the considered datasets, the memetic algorithm consistently generated the same solution in all runs (thus it is presumably optimal). By applying this method (backtracking or memetic algorithm depending on the number of vertices) recursively, we have at each step a graph as input, and the two subgraphs induced by each of the sets of the vertex partition as output; stopping when we arrive to a graph with just one vertex, we generate a hierarchical clustering in a top-down fashion.

The rationale of the use of our objective function can be clear if we rearrange its terms. We can write

$$F = \frac{A_{out}}{H_{in}}(|E| - |E_{out}|)|E_{out}| \tag{2}$$

where A_{out} is the arithmetic mean of the weights that connect vertices of S with $V \setminus S$ (the cut); H_{in} is the harmonic mean of the weights of the edges not in the cut, and $|E_{out}|$ is the cardinality of the cut. Informally, maximizing F is equivalent to try to find a cut that discriminates well the two groups, normalized by the harmonic mean of the intra-cluster dissimilarity, and multiplied by a factor that is maximum when the two groups have a similar number of elements. Normalizing by the harmonic mean allows the denominator being more stable to the presence of outlier samples when associated to either V or $V \setminus S$. For this reason, we denote this partition as *arithmetic-harmonic cut.*

Notice that maximizing the first part of the objective function, i.e., $\sum_{e\in E_{out}} w(e)$ (the total weights of edges across the two sets) is the same as solving the Max-Cut problem for graph G, which is a NP-hard problem. However, it turns out that the hierarchy generated by partitions using Max-Cut does not corroborate the previous knowledge about the datasets. This is probably due to the fact that no importance is given in Max-Cut to the similarity of vertices within the sets. We also considered the objective function

$$F' = \sum_{e\in E_{out}} w(e) - \sum_{e\in E_{in}} w(e) \tag{3}$$

However, the resulting partition by maximizing F' turns out to be no better than the partition obtained from Max-Cut.

2.2 Ultrametric Trees

Ultrametric trees constitute a very amenable approach for fitting distance matrices to trees. In essence, an ultrametric tree T is a weighted tree in which the distance D_{ij} between any two leaves i and j (measured as the sum of the weights of the edges that have to be traversed to reach i from j inside T) verifies that $D_{ij} \leqslant \max\{D_{ik}, D_{jk}\}$, $1 \leqslant i, j, k \leqslant n$, where n is the number of leaves. This equation implies that given any internal node h in T, it holds that $D_{hi} = D_{hj}$ for any leaves i, j having h as ancestor.

The use of ultrametric trees has several advantages in hierarchical classification. First of all, edge weights are very easy to compute: given a distance matrix M containing dissimilarity values for a collection of objects, and a candidate tree T, the minimum weights such that $D_{ij} \geqslant M_{ij}$ and T is ultrametric can be computed in $O(n^2)$ [5]. Secondly, they adapt very well to dynamical processes evolving at a more or less constant rate. Finally, even if the latter is not the case, they provide a very good approximation to more relaxed criteria such as mere additivity, that would be much more computationally expensive to calculate. Notice also that finding the optimal topology T for a given distance matrix M under the ultrametric assumption is NP-hard [5].

Ultrametric trees have been computed using an evolutionary two-phase procedure: firstly, a collection of high quality tentative trees are generated; subsequently, a consensus method is used to summarize this collection into a single tree. Beginning with the former, the generation of high quality (i.e., minimum weight) ultrametric trees has been approached using an evolutionary algorithm based on the scatter search template. Starting from the solution provided by the complete-link agglomerative algorithm, an initial population of trees is produced by perturbation (internal exchanges of branches). Then, an evolutionary cycle is performed using tree-based path relinking for recombination [6], and internal rotations for local search (no mutation is used). Whenever the system stagnates, the population is restarted by keeping the best solution and generating new trees by exchanging branches among existing trees.

Once a collection of high quality trees has been found, the consensus method is used to amalgamate them. This is done using the TreeRank measure [7] as similarity metric among trees. This measure is based on counting the number of times we have to traverse an edge upwards or downwards in order to go from a certain leaf to another one. By computing how different these figures are for two trees, we obtain a dissimilarity value. The TreeRank measure is currently being used in TreeBASE[1] –one of the most widely used phylogenetic databases– for the purposes of handling queries for similar trees.

The consensus algorithm we have used is an evolutionary metaheuristic that evolves tentative trees following [8]. Given the collection of trees we want to summarize, the sum of dissimilarities to the tentative tree is used as the fitness function (to be minimized). Evolution is performed using the prune-delete-graft operator [9, 10] for recombination, no mutation, binary tournament selection, and elitist replacement. In our experiments, we have considered all different trees generated by the scatter search method in one hundred runs, and then running the consensus algorithm one hundred times on this collection. The best solution out of these latter 100 runs is kept as the final consensus tree.

[1] `http://www.treebase.org`

3 Classifying Indo-European Languages

Two major hypotheses exist about the origin of Indo-European languages: the *'Kurgan expansion'* and the *'Anatolian farming'* hypotheses. Based on archaeological evidences, the Kurgan theory [11] says that Kurgan horsemen (near current Russia) went to Europe and the Near East around 6000 years B.C. On the other hand, the Anatolian theory [12] claims that Indo-European languages expanded with the spread of agriculture from Anatolia (present Turkey) around 8000-9500 years B.C. Scientists have used genetic [13, 14, 15] and numerical [16, 17] methods to complete the linguistic phylogeny of the Indo-European languages. However, there are still some doubts about its exact topology.

The dataset we analyse is the distance-matrix for 84 Indo-European languages generated by Dyen *et al.* [18]. They used a list of basic vocabulary and estimated historical relationships (similarity) between two languages by computing the ratio of number of words (cognates) shared by them and the total number of words. Furthermore, one also considers the replacement rates of each word in the vocabulary. By considering the above ratio and replacement rates, they generated the so-called "separation time" between pairs of languages, which they provided as a distance-matrix of 84 languages. This dataset is the same that was used in [17] where the *neighbor-net analysis* method provided some hints on possible lateral information transfer while still provide an overall hierarchical relationships among the languages.

Gray and Atkinson used a Bayesian Markov chain Monte Carlo method to generate and analyze a large number of (10000) tentative trees [16]. Although their method is in remarkable agreement with the timing of the Anatolian hypothesis, the method also shows how much uncertainty still exists to validate some subtrees. The posterior probability of some subtrees can be as low as 0.36 in some cases, e.g., in dividing a sub-tree into Indo-Iranian and Albanian languages, and 0.4 in dividing the Greek and Armenian groups from the most of the languages (with the exception of the Tocharian and the Hittite, extinct languages introduced by authors to the original dataset by Dyer *et al.*) [18].

We have applied our method of arithmetic-harmonic cut to Dyen *et al.*'s dataset. The language tree we have obtained using this cut is shown in Fig. 1(a). As it can be seen, this method separates the relatively older languages (a Greco-Armenian-Albanian-Indo-Iranian group) from the relatively newer languages (a Celtic-Slavic-Germanic-Romanic group). Unlike our tree, the tree in [16] (see Fig. 1(b)) first separated the extinct Tocharians, Hittite, and the Greco-Armenian group languages from the rest. Their tree had the Albanian branch together with the Indo-Iranian group. We note that the resulting subtree of Indo-Iranian-Albanian languages are divided with only 0.36 bayesian posterior probability. This raises a concern, as it may be the case that Indo-Iranian-Albanian languages are older than what was claimed in [16] and they are more suited to be temporally closer to the Greco-Armenian languages. In fact, the work in [17] with neighbor-net analysis has produced a network with Albanians very close to the Greco-Armenian languages. Thus, our topology for the main divisions seem reasonable and is in closer relationship with the spread of farming from

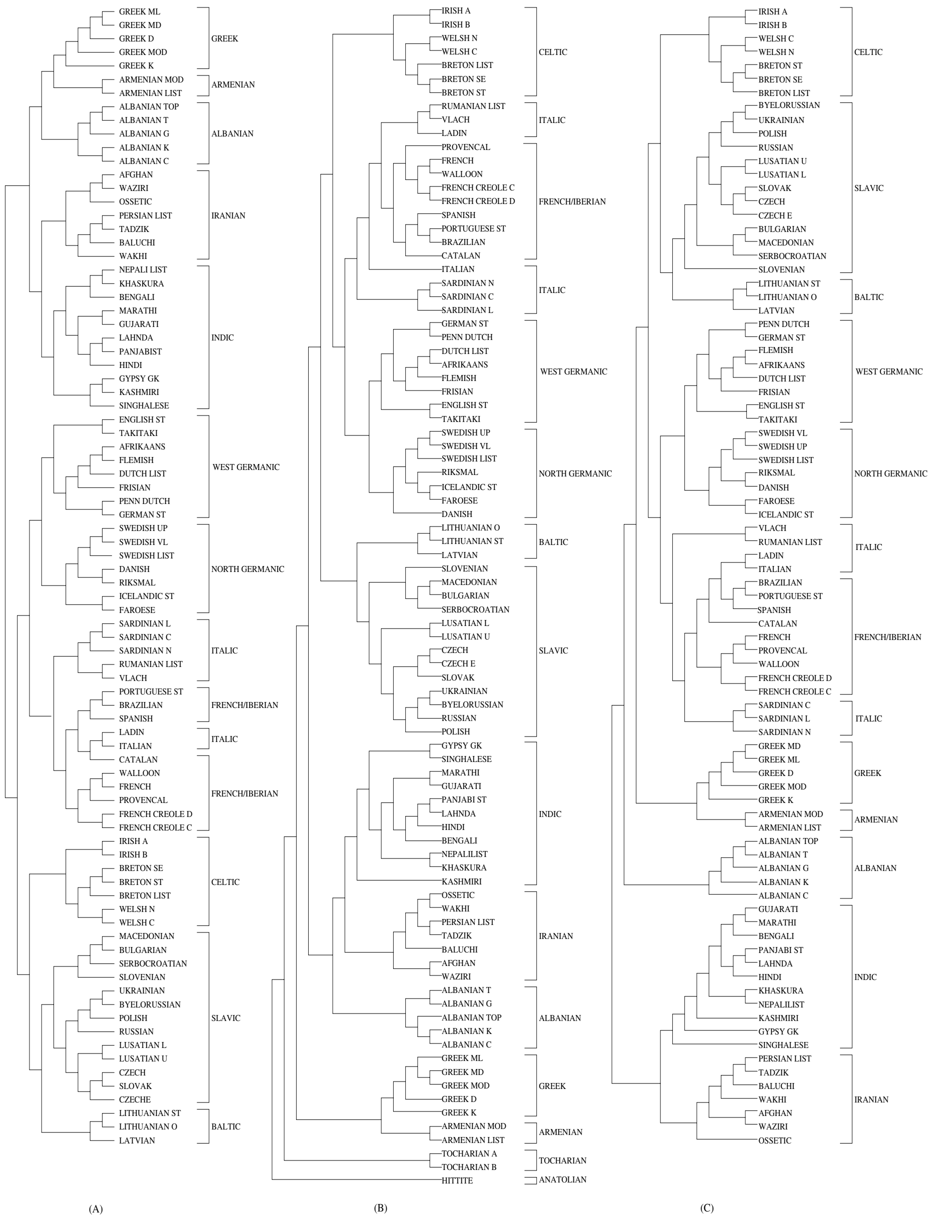


Fig. 1. Three proposed language-Trees: (a) tree using arithmetic-harmonic cuts, (b) Gray-Atkinson's tree [16], (c) consensus ultrametric tree

the Middle East between 10,000 and 6,000 years ago, which also correlates well with the first principal component of the genetic variation of 95 polymorphisms [19], which solely accounts for 28 % of the total variation.

In addition to this fact, there are certain strange relations between languages at the leaves of Gray's tree. After checking the distance matrix, we find several cases in which our tree seems to produce a more reasonable branching than Gray and Atkinson's. First of all, the closest neighbor of Czech and Slovak languages are Lusatian languages. It is probably natural to have Czech, CzechE and Slovak placed in a subtree closer to Lusatian languages. In the trees generated from both arithmetic-harmonic cut (Fig. 1(a)) and the ultrametric trees (Fig. 1(c)), we see that these languages are placed next to each other. However in Fig. 1(b) generated by [16], Czech and Slovak are placed closer to Ukrainian, Byelorussian and Russian. Secondly, Catalan is a language evolved from Latin, with strong influences from French and Spanish. As a consequence of its Latin origin, Italian is the closest language of Catalan in the dataset. The position of Catalan with Italian and Ladin in our tree seems very natural, as hybridizations with French and Spanish occurred later (note that the bayesian posterior probability is 0.83 for its link with the Spanish-Portuguese group). See [16] for the details of probabilities in the Figure 1(b). Although Italian's closest language is Ladin, the latter was placed closer to RomanianList and Vlach with the posterior probability of 0.88. Also, notice the position of the Italian with 0.59 posterior probability. Finally, there are also small differences in the topology of small subtrees between our hierarchy and Gray's, namely, those regarding Dutchlist-Afrikaans-Flemish, Greek languages, Albanian languages and the position of Bengali in the Aryan languages among others. The differences seem to occur mainly where the posterior probability of one or several branchings is low.

An important difference is that in our classification the Celtic languages are considered closer to Baltic-Slavic languages. This goes against the current belief of Celtic's closeness to Romanic and Germanic languages. Note that in Gray and Atkinson's classification, the branchings of (Germanic,Romance) and (Celtic,(Germanic,Romance)) have low posterior probabilities (0.46 and 0.67, respectively). The minimum-weight ultrametric tree (see Fig. 1(c)) for this dataset also considers Celtic and Slavic languages to be the closest ones as groups. However, this tree disagrees with our tree in the primary branches. For instance, it first takes out Indo-Afghan languages as outliers, then considers Albanian and Greco-Armenian languages as outliers successively. In the tree obtained by the arithmetic-harmonic cut, all these outliers are grouped together. Notice that even at the successive branchings, the consensus ultrametric tree often produces a large number of outliers (see e.g., Indic and Iranian branches of Figure 1(c)).

4 A Molecular Classification of 60 Tumors

Validation of our methodology on the languages dataset has allowed us to have confidence in applying it in our primary problem domain, classification of cancer samples. In this section, we show how our partitioning algorithm finds subtypes of human cancers. We study a dataset from 60 tumor cell-lines used in National Cancer Institute's (NCI) screen for anti-cancer drugs. We use the gene expression of these cell lines given as a cDNA microarray with 6,830 genes for each cell-line.

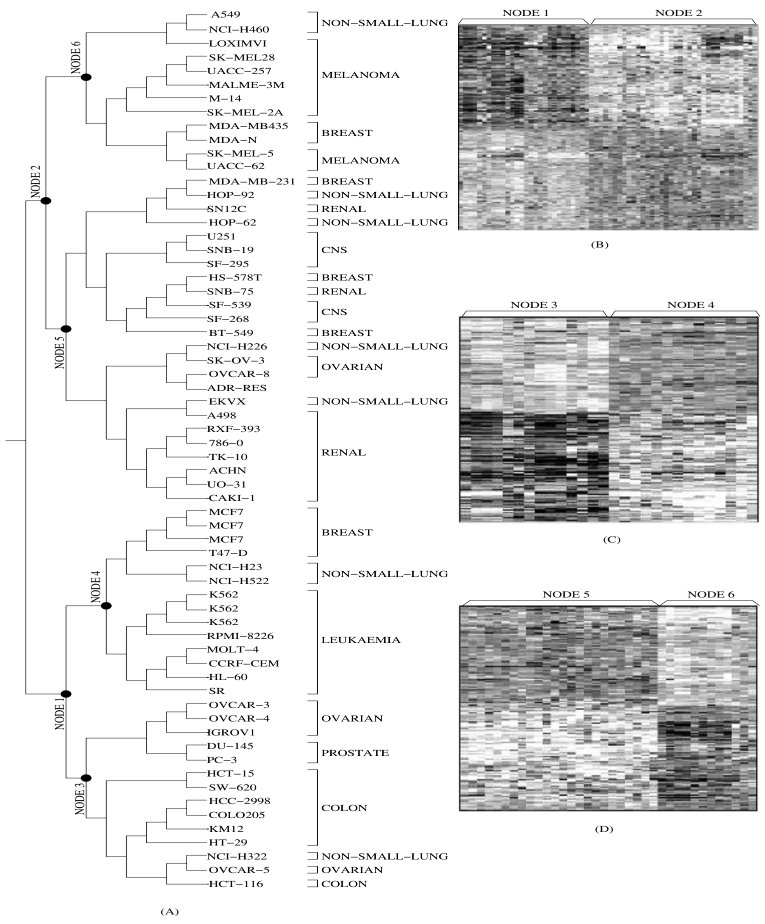


Fig. 2. (a) Classification of NCI60 dataset using arithmetic-harmonic cuts. Genetic signatures with (b) 1101 genes for the first partition into Node 1 and Node 2, (c) 695 genes for the second partition into Node 3 and Node 4, and (d) 696 genes for the third partition into Node 5 and Node 6.

The analysis of this dataset was done by Ross *et al.* in 2000 [20], where a result of a hierarchical clustering for this dataset was first discussed. Their result shows that the cell lines from same origin were grouped together in case of leukaemia, melanoma, colon, renal and ovarian cancers, with a few exceptions. However, cell-lines derived from non-small lung carcinoma and breast tumors were distributed in multiple places suggesting a heterogeneous nature.

Fig. 2(a) shows the result of applying arithmetic-harmonic cut on this dataset. In Fig. 2(b),(c) and (d), we show the genetic signatures (most differentially expressed genes in the two sides of the partition) of the first three partitions using the cut with 1,101, 696 and 695 genes respectively. In the genetic signatures (computed with the method described in [21] and [22]), each row corresponds to a gene and each column corresponds to a tumor sample.

We will now compare the hierarchy generated by arithmetic-harmonic cuts to Ross *et al.*'s classification tree and the consensus ultrametric tree, shown in Fig. 3. All clustering methods agree in the fact that some of the tumors, (e.g., leukaemia, renal, melanoma, central nervous system) are relatively homogeneous, i.e., most samples of these tumors are grouped together with a few exceptions. However, in Ross *et al.*'s solution and in the ultrametric tree, all melanoma samples except LOXIMVI are grouped with colon and leukaemia tumors (see the lower branches of both trees in Figure 3), whereas arithmetic-harmonic cut shows a marked difference in the first division. It groups *all* melanoma tumor samples (including LOXIMVI) with CNS (Central Nervous System), renal, ovarian and some of the breast and lung tumor samples (see Node 2 in Fig. 2(a)). Since CNS is the closest *group* to the melanoma samples in this figure, we may infer that this clustering supports the hypothesis that central nervous system (CNS) and melanoma may share important genetic pathways with similar expression. We note that CNS is a favorite site of metastasis in patients with advanced melanoma and that it is the first site of relapse in $15-20\%$ of melanoma cases [23, 24]. Also notice that CNS metastases from melanoma and renal tumors are also often misdiagnosed as primary brain tumors [25]. However, our literature survey did not reveal a close relationship of melanoma with colon or leukaemia, as compared to its relation with CNS.

The three methods, however, behave slightly differently in clustering non-small-lung, breast and ovarian tumors. First of all, all clustering methods applied on NCI60 group together ovarian tumor samples OVCAR-3, OVCAR04 and IGROV1. However, the positions of OVCAR-5, SK-OV-3 and OVCAR-8 differ in the outcomes of the different methods suggesting a possible heterogeneity of ovarian tumors. Also, it was suggested by Perou *et al.* [26] that there are 6 types of breast-cancers. All clustering methods more or less agree with the fact that the breast tumors (HS-578T, BT-549, MDA-MB-231, MCF7, T-47D, MDA-MB435, MDAN) are scattered in 4-5 places. In all methods, breast tumors HS-578T, BT-549 and MDA-MB231 are together with CNS/renal tumor samples; MCF7 and T-47D tumors are clustered with colon tumors; MDA-N and MDA-MB435 tumors are grouped with melanoma tumor samples. This is a definite indication that the breast cancer is a heterogeneous disease. Similarly, small-lung-cancer samples are distributed in all the three methods.

In the above comparison, we need to remember that ultrametric trees are largely used in cladistics, and assume that all species evolve at a constant rate (cf. Sect. 2). Such an assumption suits rather well the universe of languages (they evolve according to mutual contact between sub-populations, assimilating the words or phonemes from each other). However, unlike biological beings like bacteria or more complex life forms, the tumor cell-lines do not have a common ancestor. Tumors are defects in DNA, which cause malignant cell proliferation. Thus, the ultrametric approach may be susceptible to errors in the classification of cancer samples. Therefore, the position of melanoma samples in the tree produced by the arithmetic-harmonic cut should not be considered incorrect. Actually, the position of melanoma with CNS suggests an interesting quest, that

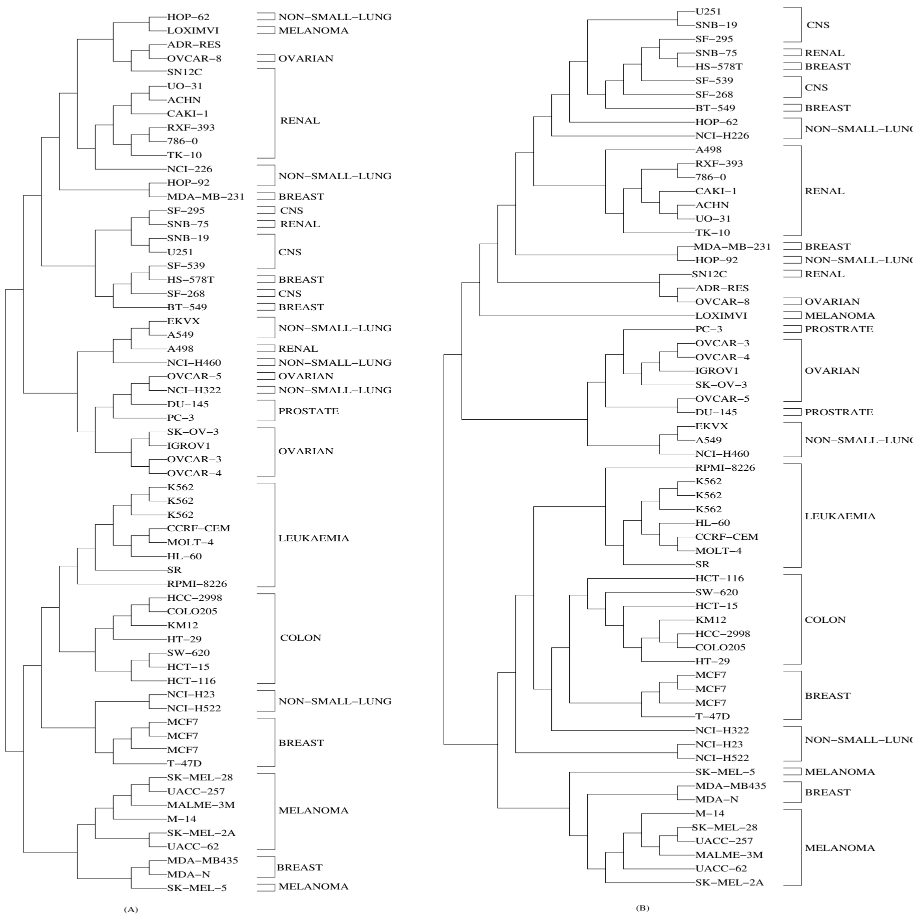


Fig. 3. Classification of NCI60 Dataset from (a) Ross *et al.* and (b) ultrametric tree

of finding common activated genetic pathways, as it is known that the brain is a primary point of metastasis to an individual with melanoma condition [23, 24].

5 Conclusions

We proposed two new approaches for hierarchical clustering and showed the result of applying these methods on two very diverse datasets. The hierarchies we produce for both languages and cancer samples in this method agree very well with existing data about these datasets. It also raises some interesting questions. The arithmetic-harmonic cut seems to correlate well with the results of the first component of the genetic variation provided by Cavalli-Sforza and his co-authors [19]. It indicates a branching in two major groups, with an earlier group "moving" towards Europe (actually, the advanced farming hypothesis at work), later followed by another group moving in the same direction (evolving in Greek and Albanian and Armenian languages) while another group "moved" southeast and later differentiated in Iranian and Indic languages. It also suggest a commonality of Celtic, Baltic and Slavic (a hypothesis also raised in the past, and also supported by the consensus of the ultrametric trees). These differences, as well as small others, with the solution provided by Gray and Atkinson's seem to be in branchings where the bayesian posterior probability is low, and our methods agree where the posterior probability is high. The consensus of the ultrametric trees seem to suggest a single wave towards Europe, but a first branching in an Albanian group, followed by a second branching with the Greek and Armenian in one subgroup seems less plausible to us.

Overall, our results seem to indicate that it is important to use several hierarchical clustering algorithms and to analyze common subgroupings. In the case of tumor samples, it is indeed the case that this is the most relevant outcome as we do not have any guarantee that the samples "share" a common "ancestor". The question: *"Which tree is the best one ?"* might actually be highly irrelevant to the the real problem at hand, as it seems to be the consensus of these trees the most important outcome. Results on a number of other clustering algorithms on these datasets (which we were unable to show here for reasons of space), indicates that more research in robust algorithm methods needs to be done for molecular subtype classification in cancer and that validation of the methodology with different problem settings is highly beneficial to develop it.

References

1. Cotta, C., Moscato, P.: A memetic-aided approach to hierarchical clustering from distance matrices: Application to phylogeny and gene expression clustering. Biosystems **71** (2003) 75–97
2. Merz, P., Freisleben, B.: Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. Evolutionary Computation **8** (2000) 61–91
3. Battiti, R., Bertossi, A.: Differential greedy for the 0-1 equicut problem. In: Proc. of DIMACS Workshop on Network Design: Connectivity and Facilities. (1997)
4. Festa, P., Pardalos, P., Resende, M.G.C., Ribeiro, C.C.: Randomized heuristics for the MAX-CUT problem. Optimization Methods and Software **7** (2002) 1033–1058
5. Wu, B., Chao, K.M., Tang, C.: Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices. Journal of Combinatorial Optimization **3** (1999) 199–211
6. Cotta, C.: Scatter search with path relinking for phylogenetic inference. European Journal of Operational Research **169** (2006) 520–532
7. Wang, J., Shan, H., Shasha, D., Piel, W.: Treerank: A similarity measure for nearest neighbor searching in phylogenetic databases. In: Proceedings of the 15th International Conference on Scientific and Statistical Database Management, Cambridge MA, IEEE Press (2003) 171–180
8. Cotta, C.: On the application of evolutionary algorithms to the consensus tree problem. In Gottlieb, J., Raidl, G., eds.: Evolutionary Computation in Combinatorial Optimization. Volume 3248 of Lecture Notes in Computer Science., Berlin, Springer-Verlag (2005) 58–67

9. Moilanen, A.: Searching for the most parsimonious trees with simulated evolution. Cladistics **15** (1999) 39–50
10. Cotta, C., Moscato, P.: Inferring phylogenetic trees using evolutionary algorithms. In Merelo, J., et al., eds.: Parallel Problem Solving From Nature VII. Volume 2439 of Lecture Notes in Computer Science. Springer-Verlag, Berlin (2002) 720–729
11. Mallory, J.P.: Search of the Indo-European languages. Archaelogy and Myth (1989)
12. Renfrew, C.: Time-depth in historical linguistics. The McDonald Institute for Archaeological Research (2000) 413–439
13. Richards, M.: Tracing european founder lineage in the near easter mtDNA pool. Am. K. Hum. Genet **67** (2000) 1251–1276
14. Semoni: The genetic legacy of Paleolithic Homo Sapiens in extant europeans: a Y chromosome perspective. Science **290** (2000) 1155–1159
15. Chikhi, L., Nichols, R., Barbujani, G., Beaumont, M.: Y genetic data support the Neolithic demic diffusion model. Prod. Natl. Acad., Sci. **67** (2002) 11008–11013
16. Gray, R.D., Atkinson, Q.D.: Language-tree divergence times support the anatolian theory of indo-european origin. Nature **426** (2003) 435–439
17. Bryant, D., Filimon, F., Gray, R.: Untangling our past: Languages, trees, splits and networks. In Mace, R., Holden, C., Shennan, S., eds.: The Evolution of Cultural Diversity: Phylogenetic Approaches. UCL Press (2005) 69–85
18. Dyen, I., Kruskal, J.B., Black, P.: An Indo-European classification: A lexicostatistical experiment. Transactions of the American Philosophical Society, New Ser. **82** (1992) 1–132
19. Cavalli-Sforza, L.: Genes, peoples, and languages. Proceedings of the National Academy of Sciences of the United States of America **94** (1997) 7719–7724
20. Ross, D.T., Scherf, U., Eisen, M., Perou, C., Rees, C., Spellman, P., Iyer, V., Jeffrey, S., Rijn, M., Waltham, M., Pergamenschikov, A., Lee, J.C., Lashkari, D., Shalon, D., Myers, T., Weinstein, J.N., Botstein, D., Brown, P.: Systematic variation in gene expression patterns in human cancer cell lines. Nature Genetics **24** (2000) 227–235
21. Cotta, C., Langston, M., Moscato, P.: Combinatorial and algorithmic issues for microarray data analysis. In: Handbook of Approximation Algorithms and Metaheuristics. Chapman and Hall (2005)
22. Hourani, M., Mendes, A., Berretta, R., Moscato, P.: A genetic signature for parkinsons disease using rodent brain gene expression. In Keith, J., ed.: Bioinformatics. Humana Press (2006)
23. Ferraresi, V., Ciccarese, M., Zeuli, M., Cognetti, F.: Central system as exclusive site disease in patients with melanoma: treatment and clinical outcome of two cases. Melanoma Res. 2005 **15** (2005) 467–469
24. Marchetti, D., Denkins, Y., Reiland, J., Greiter-Wilke, A., Galjour, J., Murry, B., Blust, J., Roy, M.: Brain-metastatic melanoma: a neurotrophic perspective. Pathology Oncology Research **9** (2003) 147–158
25. Buell, J., Gross, T., Alloway, R., Trofe, J., Woodle, E.: Central nervous system tumors in donors: Misdiagnosis carries a high morbidity and mortality. Transplantation Proceedings **37** (2005) 583–584
26. Perou, C.M., Jeffrey, S.S., Rijn, M., Rees, C.A., Eisen, M.B., Ross, D.T., Pergamenschikov, A., Williams, C.F., Zhu, S.X., Lee, J.C.F., Lashkari, D., Shalon, D., Brown, P.O., Botstein, D.: Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. Genetics **96** (1999) 9212–9217

Robust SVM-Based Biomarker Selection with Noisy Mass Spectrometric Proteomic Data

Elena Marchiori[1], Connie R. Jimenez[2],
Mikkel West-Nielsen[3], and Niels H.H. Heegaard[3]

[1] Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
elena@cs.vu.nl
[2] Department of Molecular and Cellular Neurobiology,
Vrije Universiteit Amsterdam, The Netherlands
connie.jimenez@falw.vu.nl
[3] Department of Autoimmunology, Statens Serum Institut, Copenhagen, Denmark
MWN@ssi.dk, NHE@ssi.dk

Abstract. Computational analysis of mass spectrometric (MS) proteomic data from sera is of potential relevance for diagnosis, prognosis, choice of therapy, and study of disease activity. To this aim, feature selection techniques based on machine learning can be applied for detecting potential biomarkes and biomaker patterns. A key issue concerns the interpretability and robustness of the output results given by such techniques. In this paper we propose a robust method for feature selection with MS proteomic data. The method consists of the sequentail application of a filter feature selection algorithm, RELIEF, followed by multiple runs of a wrapper feature selection technique based on support vector machines (SVM), where each run is obtained by changing the class label of one support vector. Frequencies of features selected over the runs are used to identify features which are robust with respect to perturbations of the data. This method is tested on a dataset produced by a specific MS technique, called MALDI-TOF MS. Two classes have been artificially generated by spiking. Moreover, the samples have been collected at different storage durations. Leave-one-out cross validation (LOOCV) applied to the resulting dataset, indicates that the proposed feature selection method is capable of identifying highly discriminatory proteomic patterns.

1 Introduction

Feature selection (FS) for classification can be formulated as a combinatorial optimization problem: finding the feature set maximizing the predictive performance of the classifier trained from these features. FS is a major research topic in supervised learning and data mining [10, 16, 12]. For the sake of the learning performance, it is highly desirable to discard irrelevant features prior to learning, especially when the number of available features significantly outnumbers the number of samples, like in biomedical studies. Because of its computational intractability, the FS problem has been tackled by means of heuristic algorithms based on statistics and machine learning [10, 20, 22]. Biological experiments from laboratory technologies like microarray and mass spectrometry techniques, generate data with a very high number of variables (features), in general much larger than the number of samples. Therefore FS provides a fundamental step in the analysis of such type of data [27]. Ideally one would like to detect potential biomarkers and biomarker patterns, that both highly discriminate diseased from healthy samples and are biological interpretable. However, as substantiated in recent publications like [19, 3, 21], reliability and reproducibility of results depend on the particular way samples are handled [26], on the instability of the laboratory technology, as well as on the specific techniques employed in the computational analysis.

In this paper we consider FS for classification with MS proteomic data from sera. Various machine learning and statistical techniques for feature selection have been applied to proteomic data, like [15, 17, 8, 13, 5, 6, 7], in order to detect potential tumor biomarkers for (early) cancer diagnosis (clinical proteomics). A summary of actual challenges and critical assessment of clinical proteomics can be found, e.g., in [21].

Here we propose a new method for FS with MS proteomic data. The goal is to identify potential biomarker patterns that not only highly discriminate diseased and healthy samples, but also are robust with respect to perturbation of the data. The method consists of three main steps. First, a popular filter feature selection algorithm, RELIEF, is used as pre-processing in order to reduce the number of considered features. Next, multiple runs of linear SVM are considered, where at each run a perturbed training set is used, obtained by changing the class label of one support vector. Each run generates a large subset of selected features. The frequency (over the runs) of selection of the features is used to choose the most robust ones, namely those with highest frequency. Finally, the resulting features are transformed into feature intervals, by considering the ordering of the features, where neighbour features refer to peptides of similar masses.

The method generates a subset of feature intervals, where both the number of intervals and features are automatically selected. These intervals describe potential biomarker patterns.

We analyze experimentally the performance of the method on a real-life dataset with controlled insertion of noisy samples (long storage time samples) and "relevant" features (spiked molecules) [26]. The results indicate that the method performs robust feature selection, by selecting features corresponding to m/z measurements near to the (average of m/z values of the peak of the) spiked molecules, and by misclassifying only 1 noisy sample (with long storage time).

2 Background

This section describes in brief the Machine Learning techniques we use in the proposed feature selection method.

2.1 Linear Support Vector Machines

In linear SVM-based binary classification [25, 2], samples of two classes are linearly separated by means of a maximum margin hyperplane, that is, the hyperplane that maximizes the sum of the distances between the hyperplane and its closest points of each of the two classes (the margin). When the classes are not linearly separable, a variant of SVM, called soft-margin SVM, is used. This SVM variant penalizes misclassification errors and employs a parameter (the soft-margin constant C) to control the cost of misclassification.

Training a linear SVM classifier amounts to solving the following constrained optimization problem:

$$min_{\mathbf{w},b,\xi_k} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i \qquad \text{s.t.} \qquad \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 - \xi_i$$

with one constraint for each training sample $\mathbf{x}_i$. Usually the dual form of the optimization problem is solved:

$$min_{\alpha_i} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{m} \alpha_i$$

such that $0 \leq \alpha_i \leq C$, $\sum_{i=1}^{m} \alpha_i y_i = 0$. SVM requires $O(m^2)$ storage and $O(m^3)$ to solve.

The resulting decision function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ has weight vector $\mathbf{w} = \sum_{k=1}^{m} \alpha_k y_k \mathbf{x_k}$. Samples $\mathbf{x}_i$ for which $\alpha_i > 0$ are called *support vectors*, since they uniquely define the maximum margin hyperplane. Samples with $\alpha_i - C$ are misclassified.

Maximizing the margin allows one to minimize bounds on generalization error. Because the size of the margin does not depend on the input dimension, SVM are robust with respect to data with high number of features. However, SVM are sensitive to the presence of (potential) outliers, (cf. [11] for an illustrative example) due to the regularization term for penalizing misclassification (which depends on the choice of C).

2.2 Variable Selection Techniques

One can distinguish three main approaches for feature ranking/selection: wrapper, filter and embedded.

– In the wrapper approach features are selected/ranked by taking into account their contribution to the performance of a given type of classifier (e.g., SVM).
– In the filter approach the selection/ranking of features is not (directly) biased towards the performance of a specific type of classifier, but is based on an evaluation criterion for quantifying how well feature (subsets) discriminate the two classes.
– Finally, in the embedded approach feature selection/ranking is part of the training procedure of a classifier, like in decision trees.

The effectiveness of these approaches depends on the application domain. The wrapper approach is favoured in many works since the selection of the features is directly linked to the performance of a chosen type of classifier. On the other hand, algorithms based on the filter approach, like RELIEF, are in general more efficient and have been successfully applied to real life domains [28]. Techniques based on the embedded approach provide a global approach, where feature selection is a by-product of the training algorithm for classification.

SVM Feature Selection (SVMFS)

The weights w_i of a linear SVM classifier provide information about feature relevance, where a bigger weight value implies higher feature relevance. In this paper a feature x_i is scored by means of w_i^2 [11]. Feature weights, obtained by training a linear SVM on the training set, are used in a scoring function for ranking features as described above. The algorithm for feature selection based on SVM is illustrated below (in pseudo-code).

```
SVMFS
%input: training set X, number of features
              to be selected M
%output: subset Selected of M features
train linear classifier with SVM on X;
score features using the squared value of
      the weights of the classifier;
Selected = M features with highest score;
return Selected;
```

RELIEF

RELIEF [23, 14] is a filter-based feature ranking algorithm that assigns a score to features based on how well the features separate training samples from their nearest neighbours from the same and from the opposite class.

The algorithm constructs iteratively a weight vector, which is initially equal to zero. At each iteration, RELIEF selects one sample, adds to the weight the difference between that sample and its nearest sample from the opposite class (called nearest miss), and subtracts the difference between that sample and its nearest neighbour from the same class (called nearest hit). The iterative process terminates when all training samples have been considered. The resulting weight of a feature is divided by its range of values (computed using only the training set). Subsampling can be used to improve efficiency in case of a large training set. The pseudo-code of the RELIEF algorithm used in our experiments is given below.

```
RELIEF
%input: training set X
%output:  Ranking of features
nr_feat = total number of features;
```

```
weights = zero vector of size nr_feat;
for all samples exa in training set do
    hit(exa) = nearest neighbour of exa
             from same class;
    miss(exa) =  nearest neighbour of exa
             from opposite class;
    weights =  weights-abs(exa-hit(exa))+
             abs(exa - miss(exa));
end;
scale each weight using range of corresponding m/z
value intensity over the training set;
Ranking =  obtained by sorting weights
           in decreasing order;
return Ranking;
```

3 Mass Spectrometric Proteomic Data

The MS proteomic dataset here considered is obtained by matrix-assisted laser desorption/ionization time-of-flight mass spectrometry (MALDI-TOF MS), a recent laboratory technology which offers protein profiling at high resolution and

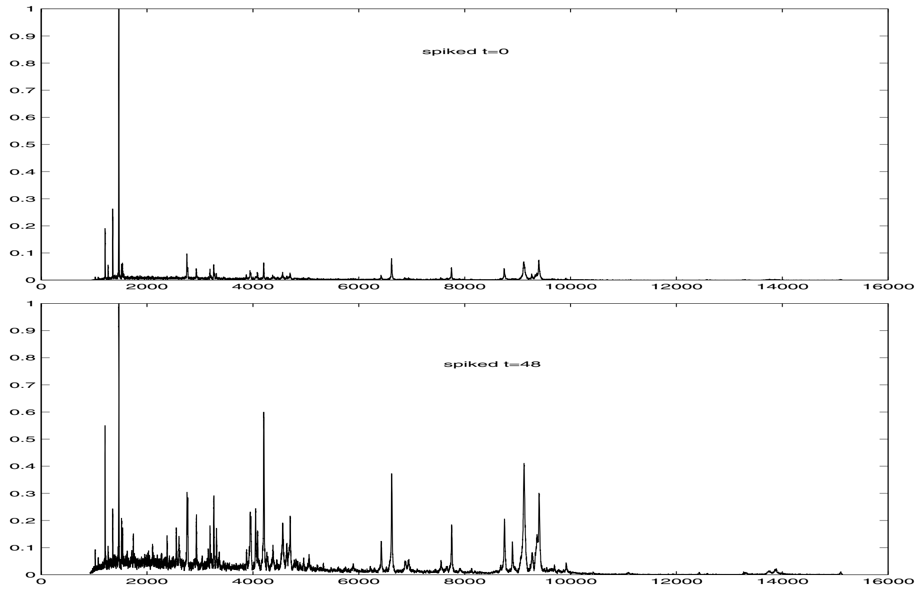


Fig. 1. A MALDI-TOF MS spiked sample for one person at storage duration time t=0 (top) and t=48 (bottom): x-axis contains (identifiers of) the m/z values of peptides and the y-axis their concentration

throughput. It measures the relative abundance of ionisable low molecular weight peptides in complex mixtures, like serum (cf. e.g. [4]). Because it is relatively inexpensive and noninvasive, it is considered a promising new technology for classifying disease status and for tumor biomarker detection.

MALDI-TOF MS technology produces a graph of the relative abundance of ionized peptides (y-axis) versus their mass-to-charge (m/z) ratios (x-axis). (see Figure 1) The m/z ratios are proportional to the peptide masses, but the technique is not able to identify individual peptides, because different peptides may have the same mass and because of limitations in the m/z resolution.

Given proteomic profiles from healthy and diseased individuals, the goal is to build a classifier for tumor diagnostics and to identify those proteins that are potentially involved in the disease.

The dataset considered in this study consists of 96 samples obtained from human sera of 8 adult persons. Spiking has been used to produce two classes, and 6 different storage durations (t=0, 1, 4, 8, 24, 48 hours) have been used to produce noisy data. Each profile contains 22572 m/z measurements. Adjacent m/z measurements correspond to peptides with similar mass versus charge. Thus the ordering on the x-axis has a biological meaning. This ordering will be used in the FS method described in the next section.

The complete procedure for generating such data is described in detail in [26], where this and other datasets have been analyzed for the first time. Calibration standards containing seven peptides and four proteins were used as artificial markers (Bruker Daltonik) and consisted of the following molecules with average molecular masses given in parentheses: angiotensin II (1047.20), angiotensin I (1297.51), substance P (1348.66), bombesin (1620.88), ACTH clip 1-17 (2094.46), ACTH clip 18-39 (2466.73), somatostatin 28 (3149.61), insulin (5734.56), ubiquitin I (8565.89), and cytochrome c (6181.05) and myoglobin (8476.77). However, no signal was recovered for the following four spiked molecules, possibly due to losses during the laboratory sample processing procedure: substance P (1348.6), ACTH clip 1-17 (2094.46), cytochrome *c* (6181.05), and myoglobin (8476.77) [26].

In [18], we used this dataset for comparing the performance of two popular feature selection techniques, RELIEF and Recursive Feature Selection with linear SVM, and the performance of two classification techniques, SVM and K nearest neighbours. The results indicated that, in general, better predictive performance does not correspond to better biological interpretability of the selected features (m/z values).

4 The Method

We propose a FS method, consisting of three steps, called Filter, Wrapper, and Interval step (FWI). The method is illustrated below in pseudo-code.

```
FWI
%input: training set X
%number M of features to be selected by RELIEF
```

```
%number N (N<M) of features to be selected by SVMFS
%SVM parameter C
%output:  Set Int of feature intervals

%FILTER step:
F = M features selected with RELIEF;

%WRAPPER step:
SV = set of support vectors obtained by training
     SVM on X;
for x in SV
 T = X with label of x changed;
 F(x) = N features selected by SVMFS applied to T;
end;
count = maximum number of times that a feature
        occurs in the sequence of F(x), x in SV;
W = features occurring count times  in the sequence
    of F(x), x in SV;

%INTERVAL step:
Cl = {C1, .., Cn} clustering of W, with
               Ci = (w(1),.., w(ni))
               w(1)<..< w(ni)
               s.t. idx(w(j+1))-idx(w(j)) <= 2
               for all j in [1..ni];
Int = {Int_1, .., Int_n} intervals from Cl, with
    Int_i= {w in Features s.t. w >= min(Ci)
             and w<= max(Ci)} for i in [1,n];
return Int;
```

Let us explain a bit in more detail the steps performed by FWI.

– FWI starts by skimming the number of features, by applying the Filter (F) step. Here RELIEF is employed in order to select M features. In the F step one typically retains about M=5000 m/z measurements from the initial 22572.
– In the Wrapper (W) step, robust wrapper based feature selection is performed using the features that passed the Filter selection. In the Wrapper (W) step, the support vectors of SVM trained on all the features are used for perturbing the data. More precisely, multiple runs of SVMFS are performed, where at each run the class label of one support vector is changed. Each run generates a set of N features (typical value N=1000). The resulting sequence of feature sets is then considered. The maximum number *count* of times a feature occurs in the sequence is computed, and all features occurring *count* times in the sequence are selected.
– Finally, in the Interval (I) step, the selected m/z features are segmented as follows. The sequence of features in W, ordered by m/z values, is segmented

in such a way that each pair of consecutive features in one segment C_i is at most two positions apart in the sequence of all features ordered by m/z values (in the above pseudo-code $idx(w)$ gives the position of feature w in the ordered sequence of all features). Finally each sequence C_i generates one interval I_i containing all m/z measurements between the first and last element of C_i.

The F step can be viewed as a kind of preprocessing, while steps W and I are heuristics for overcoming the problem of variability due to perturbations of the data that can possibly originate from noise. The method requires the user to specify 3 parameters, in particular the sizes M and N of the feature subsets selected by RELIEF and SVMFS, respectively. However, note that the values of M and N can be chosen to be fairly big, and the final smaller size of the feature subset selected by FWI is determined automatically by the algorithm.

5 Numerical Experiments

In order to assess the effectiveness of the modules of FWI, we consider the following four algorithms:

1. Wrapper feature selection (W), obtained by applying the W step of the FWI.
2. Wrapper Interval (WI), obtained by applying steps W followed by I.
3. Feature Wrapper (FW), obtained by applying steps F followed by W.
4. The complete Feature Wrapper Interval algorithm FWI.

Because of the small size of the data, LOOCV is used for comparing the performance of the four algorithms (cf., e.g., [9]). At each leave-one-out run, all but one element of the data is used as training set, and the left-out element is used for testing the predictive performance of the resulting classifier. Observe that the 96 samples of the considered dataset are not independent one of the other, as required for a correct application of LOOCV, because they are generated from 8 persons, and neither the 6 different storage times nor the spiking guarantee the production of independent samples. Nevertheless, the corresponding bias introduced in the LOOCV procedure affects the results of each algorithm, hence the results can be used for comparing the performance of the algorithms. However, such bias possibly affects the estimation of the generalization error.

Table 1 summarizes LOOCV performance results of the experiments. We use accuracy, sensitivity and specificity as quality measures for comparing the algorithms. Other measures, like AUC (Area Under the ROC Curve), can be used. As illustrated e.g. in [1], there is a good agreement between accuracy and AUC as to the ranking of the performance of the classification algorithms.

The results indicate that there is an improvement in predictive performance of the four algorithms, with best accuracy achieved by FWI.

The misclassified samples over all the LOOCV runs have storage time equal to 24 or 48 hours, indicating that longer storage time affects negatively classification of proteomic samples. Algorithm W misclassifies a total of 5 samples, of

Table 1. Results: LOOCV sensitivity, specificity and accuracy (with standard deviation between brackets)

Method	Accuracy	Sensitivity	Specificity
W	0.9479 (0.2234)	0.9375 (0.2446)	0.9583 (0.2019)
WI	0.9583 (0.2019)	0.9583 (0.2019)	0.9583 (0.2019)
FW	1.0000 (0.0000)	0.9583 (0.2019)	0.9792 (0.1436)
FWI	1.0000 (0.0000)	0.9792 (0.1443)	0.9896 (0.1021)

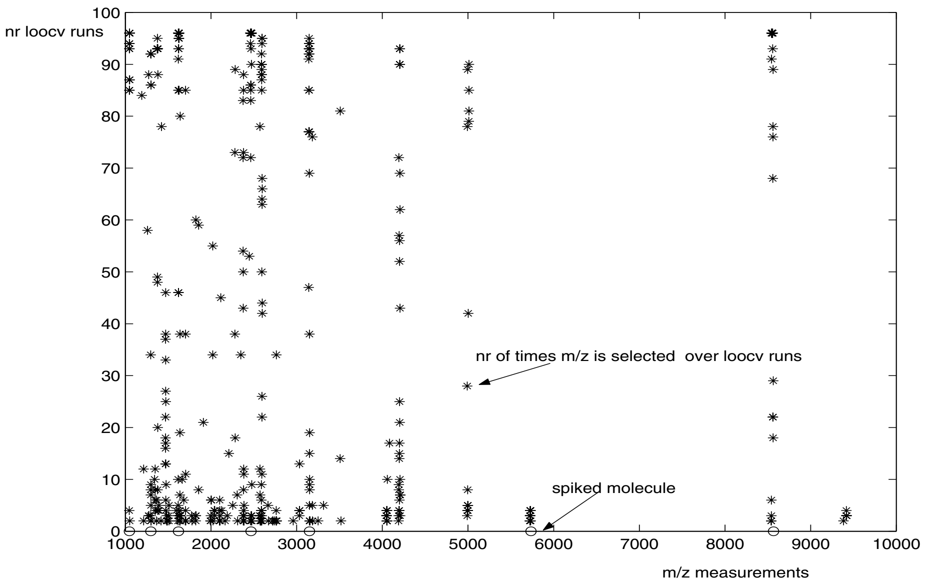


Fig. 2. Number of m/z measurements selected over LOOCV runs

which 2 spiked at t= 48, 1 spiked at t=24, and 2 normal at t=24. WI improves by correcly classifying the one spiked sample with t=24. Furthermore, FW misclassifies a total of 2 samples, the normal sample at t=24 like W and WI, and one normal at t=48, while it correctly classifies all spiked samples. Finally, FWI only misclassifies a total of 1 sample, the normal one at t=24, like W, WI, and FW.

Each algorithm selects about 120 features at each run, which are distributed (in the Interval step) in about 15 clusters.

We further analyze the results of FWI. Figure 2 shows m/z measurements versus the number of times they are selected over all LOOCV. On the x-axis the location of the spiked molecules is indicated by circles. The plot indicates that m/z

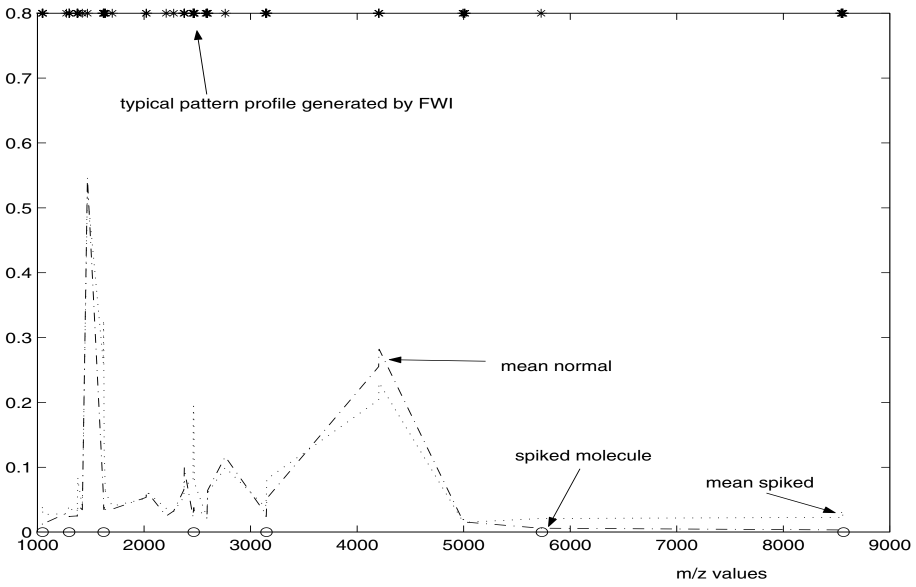


Fig. 3. A typical m/z selection generated by FWI, the corresponding values of the mean spiked and normal profile at the selected m/z values, and the spiked molecules

measurements in proximity of spiked molecules are more often selected over the LOOCV runs, except for m/z measurements in the neighbourhood of 4000 and 5000, which do not correspond to m/z measurements of spiked molecules. In the absence of additional information (e.g. tandem mass spectra yielding sequence tags) it is difficult to know what these peak values represent. One possibility is that the higher molecular weight spiked molecules are partially degraded in serum, and these peaks are proteolytically cleaved peptides from larger proteins (due to large storage time at room temperature) in the sample itself. However, this possibility has not yet been examined in depth. Figure 3 shows a typical set of m/z measurements generated by FWI, and the mean value of the intensities of spiked and normal samples for the selected m/z measurements.

In conclusion, results indicate that FWI performs robust m/z selection, where the selected features are close to the spiked molecules, and the misclassification error is close to zero, with misclassification of only noisy (that is high storage temperature) samples.

6 Conclusion

We have proposed a method for robust feature selection with MS proteomic data. The method can be considered as a small step towards the development of a feature selection methodology addressing the specific issues of the underlying laboratory technology. In particular, in this paper we addressed the issue of performing robust feature selection in the presence of noisy samples which perturb the data and negatively affect sample classification. The W and I steps of the proposed FWI method provide heuristics for tackling this problem.

This issue is related to broader questions about reproducibility and validity of results in the discovery-based "omics" research [21, 24]. In a special session on genomics of a recent issue of *Science* an essay entitled "Getting the noise out of gene arrays" noted that "[t]housands of papers have reported results obtained using gene array ... But are these results reproducible?" [19]. A controversy about reprodicibility and validity of results from MS proteomic data is ongoing [3, 21] and the path for achieving such ambitious goals appears still long.

Acknowledgments

We would like to thank the anonymous referees for their constructive comments, and Jan Rutten for stimulating discussions.

References

1. A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(6):1145–1159, 1997.
2. N. Cristianini and J. Shawe-Taylor. *Support Vector machines.* Cambridge Press, 2000.
3. E.P. Diamandis. Analysis of serum proteomic patterns for early cancer diagnosis: Drawing attention to potential problems. *Journal of the National Cancer Institute*, 96(5):353–356, 2004.
4. H.J. Issaq et al. SELDI-TOF MS for diagnostic proteomics. *Anal. Chem.*, 75(7):148A–155A, 2003.
5. Petricoin E.F. et al. Serum proteomic patterns for detection of prostate cancer. *Journal of the National Cancer Institute*, 94(20):1576–1578, 2002.
6. Petricoin E.F. et al. Use of proteomic patterns in serum to identify ovarian cancer. *The Lancet*, 359(9306):572–7, 2002.
7. Qu Y. et al. Boosted decision tree analysis of surface-enhanced laser desorption/ionization mass spectral serum profiles discriminates prostate cancer from noncancer patients. *Clin. Chem*, 48(10):1835–43, 2002.
8. Zhu W. et al. Detection of cancer-specific markers amid massive mass spectral data. *PNAS*, 100(25):14666–14671, 2003.
9. T. Evgeniou, M. Pontil, and A. Elisseeff. Leave one out error, stability, and generalization of voting combinations of classifiers. *Mach. Learn.*, 55(1):71–97, 2004.
10. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Machine Learning*, 3:1157–1182, 2003. Special Issue on variable and feature selection.
11. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.
12. George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.
13. K. Jong, E. Marchiori, M. Sebag, and A. van der Vaart. Feature selection in proteomic pattern data with support vector machines. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2004.

14. K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Tenth National Conference on artificial intelligence*, pages 129–134, 1992.
15. J. Li, Z. Zhang, J. Rosenzweig, Y.Y. Wang, and D.W. Chan. Proteomics and bioinformatics approaches for identification of serum biomarkers to detect breast cancer. *Clinical Chemistry*, 48(8):1296–1304, 2002.
16. H. Lie and editors H. Motoda. *Feature Extraction, Construction and Selection: a Data Mining Perspective.* International Series in Engineering and Computer Science. Kluwer, 1998.
17. H. Liu, J. Li, and L. Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.
18. E. Marchiori, N.H.H. Heegaard, M. West-Nielsen, and C.R. Jimenez. Feature selection for classification with proteomic data of mixed quality. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 385–391, 2005.
19. E. Marshall. Getting the noise out of gene arrays. *Science*, 306:630–631, 2004. Issue 5696.
20. Il-Seok Oh, Jin-Seon Lee, and Byung-Ro Moon. Local search-embedded genetic algorithms for feature selection. In *16 th International Conference on Pattern Recognition (ICPR'02).* IEEE Press, 2002.
21. D.F. Ransohoff. Lessons from controversy: Ovarian cancer screening and serum proteomics. *Journal of the National Cancer Institute*, 97:315–319, 2005.
22. M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):164–171, 2000.
23. L. A. Rendell and K. Kira. A practical approach to feature selection. In *International Conference on machine learning*, pages 249–256, 1992.
24. Michiels S., Koscielny S., and Hill C. Prediction of cancer outcome with microarrays: a multiple random validation strategy. *The Lancet*, 365(9458):488–92, 2005.
25. V.N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.
26. M. West-Nielsen, E.V. Hogdall, E. Marchiori, C.K. Hogdall, C. Schou, and N.H.H. Heegaard. Sample handling for mass spectrometric proteomic investigations of human sera. *Analytical Chemistry*, 11(16):5114–5123, 2005.
27. E.P. Xing. Feature selection in microarray analysis. In *A Practical Approach to Microarray Data Analysis.* Kluwer Academic, 2003.
28. L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, pages 856–863, 2003.

On the Use of Variable Complementarity for Feature Selection in Cancer Classification

Patrick E. Meyer and Gianluca Bontempi

Université Libre de Bruxelles, (CP 212), 1050 Bruxelles, Belgique
{pmeyer, gbonte}@ulb.ac.be
http://ulb.ac.be/di/mlg/

Abstract. The paper presents an original filter approach for effective feature selection in classification tasks with a very large number of input variables. The approach is based on the use of a new information theoretic selection criterion: the double input symmetrical relevance (DISR). The rationale of the criterion is that a set of variables can return an information on the output class that is higher than the sum of the informations of each variable taken individually. This property will be made explicit by defining the measure of *variable complementarity.* A feature selection filter based on the DISR criterion is compared in theoretical and experimental terms to recently proposed information theoretic criteria. Experimental results on a set of eleven microarray classification tasks show that the proposed technique is competitive with existing filter selection methods.

1 Introduction

Statisticians and data-miners are used to build predictive models and infer dependencies between variables on the basis of observed data. However, in a lot of emerging domains, like bioinformatics, they are facing datasets characterized by a very large number of features (up to several thousands), a large amount of noise, non-linear dependencies and, often, only several hundreds of samples. In this context, the detection of functional relationships as well as the design of effective classifiers appears to be a major challenge. Recent technological advances, like microarray technology, have made it possible to simultaneously interrogate thousands of genes in a biological specimen. It follows that two classification problems commonly encountered in bioinformatics are how to distinguish between tumor classes and how to predict the effects of medical treatments on the basis of microarray gene expression profiles. If we formalize this prediction task as a supervised classification problem, we realize that we are facing a problem where the number of input variables, represented by the number of genes, is huge (around several thousands) and the number of samples, represented by the clinical trials, is very limited (around several tens). Because of well-known numerical and statistical accuracy issues, it is typically necessary to reduce the number of variables before starting a learning procedure. Furthermore, selecting features (i.e. genes) can increase the intelligibility of a model while at the same time decreasing measurements and storage requirements [1]. A number of experimental studies [2, 3, 4] have shown that irrelevant and redundant features can dramatically reduce the predictive accuracy of models builded from data.

Feature selection is a topic of machine learning whose objective is selecting, among a set of input variables, the ones that will lead to the best predictive model. Two well-known approaches in feature selection combine a search strategy with a stochastic evaluation function: the *filter approach* and the *wrapper approach* (see [3, 2]). In the wrapper approach, the evaluation function is the validation outcome (e.g. by leave-one-out) of the learning algorithm itself. In the filter approach, examples of evaluation functions are probabilistic distance, interclass distance, information theoretic or probabilistic dependence measures. These measures are often considered as intrinsic properties of the data, because they are calculated directly on the raw data instead of requiring a learning model that smoothes distributions or reduces the noise.

This paper will focus on the use of filter techniques for feature selection in supervised classification tasks. In particular, we present an original filter approach based on a new information theoretic selection criterion, called the *double input symmetrical relevance* (DISR). This criterion combines two well known intuitions of feature selection: first, a combination of variables can return more information on the output class than the sum of the information returned by each of the variables taken individually. This property will be made explicit by defining the notion of *variable complementarity.* Secondly, in absence of any further knowledge on how subsets of d variables should combine, it is intuitive to assume a combination of the best performing subsets of $d-1$ variables as the most promising set. This intuition will be made formal by the computation of a lower-bound on the information of a subset of variables expressed as the average of information of all its sub-subsets.

The DISR criterion can be used to select among a finite number of alternative subsets the one expected to return the maximum amount of information on the output class. As we intend to benchmark its performance with respect to state-of-the-art information theoretic criteria we define an experimental session where several filter algorithms with different selection criteria but the same search strategy are compared. In our experiments we compare the filter based on DISR with four state-of the art approaches: a Ranking algorithm [5] and three filters based on the same search strategy: the forward selection. The three state-of-the-art criteria are the Relevance criterion [6], the Minimum Redudancy Maximum Relevance criterion [7] and the Conditional Mutual Information Maximization criterion [8]. The assessment of the different filters is obtained by measuring the classification accuracy of several learning algorithms which adopt as inputs the set of variables returned by each of the filter methods. For our benchmark purposes, we use eleven public-domain multi-class microarray gene expression datasets. The experimental results show that the proposed technique is competitive with existing filter selection methods.

2 Information Theoretic Notions for Feature Selection

This paper deals with supervised multi-class classification. We will assume either that all the variables are discrete or that they can be made discrete by a quantization step. Hereafter, we will denote by Y the discrete output random variable representing the class and by X the multi-dimensional discrete input random variable.

In qualitative terms, feature selection boils down to select, among a set of potential variables, the most *relevant* ones. At the same time it would be appealing that these selected variables are not *redundant*. The notions of relevance and redundancy can be made more formal thanks to the use of dependency measures [2, 9].

Let us first introduce some concepts of information theory:

Definition 1. *[10] The conditional entropy of Y given Z, denoted by $H(Y|Z)$ is defined as:*

$$H(Y|Z) = -\sum_{y\in\mathcal{Y}}\sum_{z\in\mathcal{Z}} p(y,z)\log p(y|z) \tag{1}$$

and the $I(X;Y|Z)$ is the conditional mutual information.

Definition 2. *[10] The conditional mutual information of the random variables X and Y given Z is defined as:*

$$I(X;Y|Z) = H(X|Z) - H(X|Z,Y) \tag{2}$$

These definitions allow us to introduce the following measure of relevance proposed by [6]:

Definition 3. Relevance.
Consider three random variables X,Y and Z and their joint probability distribution $p_{X,Y,Z}(x,y,z)$. If $H(Y|Z) = 0$, then the variable relevance of X to Y given Z, denoted by $r(X;Y|Z)$, is zero. Else if $H(Y|Z) \neq 0$, then the variable relevance of X to Y given Z is defined as:

$$r(X;Y|Z) = \frac{I(X;Y|Z)}{H(Y|Z)} \tag{3}$$

According to this definition the relevance is a function $0 \leq r(X;Y|Z) \leq 1$ that measures the relative reduction of uncertainty of Y provided by X once the value of Z is given.

In the following, we rather consider as measure of relevance the classical (non-normalized) mutual information, i.e. $I(X_i;Y|X_S)$, where X_i denotes an input variable and X_S a subset of variables not containing X_i.

The formalization of the notion of relevance makes explicit one of the major challenges in feature selection: the mutual information between an input variable X_i and the output class Y is *conditionally dependent.* This means that an input variable X_i having a significative relevance to Y given X_S, can return a null relevance conditioned on an other variable. The following two examples may serve to better illustrate this idea.

Example 1. Consider the four random variables Y, X_i, X_S and X_M such that $Y = X_i + X_S$ and $X_M = \frac{X_i}{2}$.

Given X_S, X_i has a large relevance ($I(X_i;Y|X_S) = H(Y|X_S)$), while its relevance goes to zero in the case where it is conditioned to X_M (i.e. $I(X_i;Y|X_M) = 0$). In this example, X_i and X_M have both a high mutual information with the output Y but a low conditional mutual information when conditioned to the other variable.

The next example shows that the relevance can also increase by conditioning.

Example 2. Consider the three random variables Y, X_i and X_S such that Y and X_i are independent and $Y = X_i + X_S$.

In this case the mutual information of X_i with Y is zero ($I(X_i;Y) = 0$) whereas the conditional mutual informatione increases up when conditioned to X_S (i.e. $I(X_i;Y|X_S) = H(Y|X_S)$).

These examples show that it is hard to predict, in terms of relevance, the joint effect of several input variables on an output variable Y.

As shown in Example 1 the mutual information $I(X_{i,j};Y)$ of a set $\{X_i, X_j\}$ (aka joint mutual information [11]) can be smaller than the sum of each relevance taken separately. In generic terms, we could describe these two variables as *redundant* for the task of classifying Y. For a formal definition of redundancy we refer the reader to [9, 7]. Also, as shown in Example 2, it could happen that two variables have jointly a larger mutual information with Y than when they are considered separately. In this case we say that the two variables are *complementary.* Note that *variable complementarity* should warn us against eliminating variables with null mutual information with the output (i.e. $I(X_i;Y) = 0$) since the joint information of two random variables $I(X_{i,j};Y)$ can be higher than the sum of their individual informations $I(X_i;Y)$ and $I(X_j;Y)$.

Variable complementarity was underlined experimentally in [1] and explained in [12] as a second order term of the Möbius representation of the mutual information. It can be useful to define the notion of complementarity between two variables with respect to an output Y as the difference between the joint mutual information and the sum of the "individual" mutual informations. We introduce then the following measure:

Definition 4. *The* complementarity *of two random variables X_i and X_j with respect to an output Y is defined as,*

$$C_Y(X_i, X_j) = I(X_{i,j};Y) - I(X_i;Y) - I(X_j;Y) \tag{4}$$

where $X_{i,j} = \{X_i, X_j\}$.

We define two variables as complementary if their measure of complementarity with respect to Y is positive. Note that if the complementarity is zero, the two variables are independent. We remark also that a negative value of the complementarity can be taken as a measure of the redundancy of a pair of variables for the task of predicting Y.

The example 2 is an illustration of complementarity between X_i and X_S since in that case:

$$I(X_{i,S};Y) > \underbrace{I(X_i;Y)}_{0} + I(X_S;Y) \tag{5}$$

Another illustration of complementarity is given by the well-known XOR problem [2, 1]:

Example 3. Xor problem:

X_1	1	1	0	0
X_2	1	0	1	0
$Y = X_1 \oplus X_2$	0	1	1	0

We see that X_1 and X_2 have a null mutual information with the output, once they are taken individually (i.e. $I(X_1;Y) = 0$, $I(X_2;Y) = 0$). However, when they are taken together the mutual information $I(X_{1,2};Y) = H(Y) > 0$ of the subset is positive.

Complementarity explains why an apparently irrelevant combination of variables can eventually perform efficiently in a learning task. In the following section, we will proceed to a critical survey of information theoretic approaches existing in literature, by stressing when and where the notion of complementarity is taken into account.

3 State of the Art

As mutual information can measure relevance, this quantity is currently used in literature for performing feature selection. One of the main reasons for adopting it is its low complexity computational cost ($O(d \times N)$ where d is the number of variables and N is the number of samples) in the case of discrete variables. The following sections will sketch four state-of-the-art filter approaches that use this quantity.

3.1 Variable Ranking (Rank)

The ranking method returns a ranking of variables on the basis of their individual mutual information with the output. This means that, given n input variables, the method first computes n times the quantity $I(X_i, Y)$, $i = 1, \ldots, n$, then ranks the variables according to this quantity and eventually discards the least relevant ones [5].

The main advantage of the method is its rapidity of execution. Indeed, only n computations of mutual information are required for a resulting complexity $O(n \times 2 \times N)$. The main drawback derives from the fact that possible redundancies between variables is not taken into account. Indeed, two redundant variables, yet highly relevant taken individually, will be both well ranked. As a result, a model that uses these two variables is dangerously prone to an increased variance without any gain in terms of bias reduction. On the contrary, two variables can be complementary to the output (i.e. highly relevant together) while each of them appears to be poorly relevant once taken individually (see Example 2 or Example 3). As a consequence, these variables could be badly ranked, or worse eliminated, by the ranking filter.

3.2 Filters Combining Relevance and Redundancy Analysis

Although the variable ranking algorithm is reputed to be fast, it may be poorly efficient as it only relies on individual relevance. Recently, new algorithms that combine relevance and redundancy analysis offer a good trade-off between accuracy and computational load as the Fast Correlation Based Filter [9]. Also, some heuristic search methods such as the best first search (also known as the forward selection) can be combined efficiently with information theoretic criteria in order to select the best variable given a previously selected subset.

Forward Selection is a search method that starts with an empty set of variables. At each step, it selects the variable that brings the best improvement (according to the selection criterion). As a consequence, each selected variable does influence the evaluations of the following steps. This hill-climbing search selects a subset of $d < n$ variables in d steps and explores only $\sum_{i=0}^{d}(n-i)$ evaluations.

In the following sections, several information theoretic criteria existing in the literature and that can be easily combined with the forward selection, are presented.

Relevance Criterion (REL). The relevance criterion is a well-known criterion which is used together with the forward selection search strategy [6]. The approach consists in updating a set of selected variables X_S with the variable X_i featuring the maximum relevance $I(X_i;Y|X_S)$. This strategy prevents from selecting a variable which, though relevant to Y, is redundant with respect to a previously selected one.

In analytical terms, the variable X_{REL} returned by the relevance criterion is,

$$X_{REL} = \arg \max_{X_i \in X_{-S}} \{I(X_i;Y|X_S)\} \tag{6}$$

where $X_{-S} = X \setminus X_S$ is the set difference between the original set of inputs X and the set of variables X_S selected so far[1].

Although this method is appealing, it presents some major drawbacks. The estimation of the relevance requires the estimation of several multivariate densities, a problem known to be ill-posed. For instance, at the dth step of the forward search, the search algorithm asks for $n-d$ evaluations where each evaluation requires the computation of a $(d+1)$-variate density. It is known that, for a large d, the estimations are poorly accurate and computationally expensive. For these two reasons we recently assisted to the adoption of selection criteria based on bi- and tri-variate densities only.

[1] Note that in [6] a normalized version of relevance (Eq. 3) is used.

Minimum Redundancy - Maximum Relevance criterion (MRMR). The minimum redundancy - maximum relevance criterion [7] consists in selecting the variable X_i among the not yet selected features X_{-S} that maximizes $u_i - z_i$, where u_i is a relevance term and z_i is a redundancy term. More precisely, u_i is the relevance of X_i to the output Y alone, and z_i is the mean redundancy of X_i to each variables $X_j \in X_S$ already selected.

$$u_i = I(X_i;Y) \tag{7}$$

$$z_i = \frac{1}{d} \sum_{X_j \in X_S} I(X_i;X_j) \tag{8}$$

$$X_{MRMR} = \arg \max_{X_i \in X_{-S}} \{u_i - z_i\} \tag{9}$$

At each step, this method selects the variable which has the best trade-off relevance-redundancy. This selection criterion is fast and efficient. At step d of the forward search, the search algorithm computes $n - d$ evaluations where each evaluation requires the estimation of $(d + 1)$ bi-variate densities (one for each already selected variables plus one with the output). It has been shown in [7] that the MRMR criterion is an optimal first order approximation of the conditional relevance criterion. Furthermore, MRMR avoids the estimation of multivariate densities by using multiple bivariate densities.

Note that, although the method aims to address the issue of redundancy between variables through the term z_i, it is not able to take into account the complementarities between variables. This could be ineffective in situations like the one of Example 2 where, although the set $\{X_i, X_S\}$ has a large relevance to Y, we observe that

1. the redundancy term z_i is large due to the redundancy of X_i and X_S
2. the relevance term u_i is small since X_i is not relevant to Y.

Conditional Mutual Information Maximization Criterion (CMIM). This approach [8] proposes to select the feature $X_i \in X_{-S}$ whose minimal conditional relevance $I(X_i;Y|X_j)$ among the selected features $X_j \in X_S$, is maximal. This requires the computation of the mutual information of X_i and the output Y, conditional on each feature $X_j \in X_S$ previously selected. Then, the minimal value is retained and the feature that has a maximal minimal conditional relevance is selected.

In formal notation, the variable returned according to the CMIM [2] criterion is,

$$X_{CMIM} = \arg \max_{X_i \in X_{-S}} \{ \min_{X_j \in X_S} I(X_i;Y|X_j)\} \tag{10}$$

This selection criterion is powerful. It selects relevant variables, it avoids redundancy, it avoids estimating high dimensional multivariate densities and unlike the previous method, it does not ignore variable complementarity. However,

[2] Note that in [8] this method was applied to select binary features in a pattern recognition task.

it will not necessary select a variable complementary with the already selected variables. Indeed, a variable that has a high complementarity with an already selected variable will be characterized by a high conditional mutual information with that variable but not necessarily by a high minimal conditional information (see example 3).

In terms of complexity, note that at the dth step of the forward search, the algorithm computes $n - d$ evaluations where each evaluation following CMIM requires the estimation of d tri-variate densities (one for each previously selected variable).

In the following chapter, we propose a new criterion that deals more explicitly with complementary variables.

4 Double Input Symmetrical Relevance (DISR) Criterion

A lower bound on mutual information. In this section, we derive a lower bound on the the mutual information between a subset X_S and a target variable Y. It is shown [13] that this quantity is lower bounded by the average of the same quantity computed for all the sub-subsets $X_{S-i} = X_S \setminus X_i$ of X_S.

Theorem 1. *Let $X_S = \{X_1, ..., X_d\}$ be a subset of d variables of X and $X_{S-i} = X_S \setminus X_i$, $i \in 1, \dots, d$ a subset of X_S that does not contain the variable X_i. Then,*

$$I(X_S;Y) \geq \frac{1}{d} \sum_{i \in S} I(X_{S-i};Y) \tag{11}$$

The theorem expresses that the mutual information of a subset S and a target variable Y is lower bounded by the quantity $\mathcal{L}\left(I(X_S;Y)\right)$, that is the average of the same quantity computed for all the sub-subsets X_{S-i} of X_S.

In the following, we will use this theorem as a theoretical support to the following heuristic: *without any additional knowledge on how subsets of d variables should combine, the most promising subset is a combination of the best performing subsets of ($d - 1$ variables).*

Criterion. Given a fixed number d of variables, we can write the problem of feature selection in the following form:

$$S_{best} = \arg \max_{S \,:\, |S|=d} I(X_S;Y) \tag{12}$$

In other words, the goal of feature selection is to find the subset of d variables which maximizes the mutual information with the output Y.

Our idea consists in replacing the maximization of the quantity $I(X_S;Y)$ by the maximization of its lower bound $\mathcal{L}\left(I(X_S;Y)\right)$:

$$\arg \max_{S \,:\, |S|=d} I(X_S;Y) \geq \arg \max_{S \,:\, |S|=d} \sum_{i \in S} I(X_{S-i};Y) \tag{13}$$

Replacing again the right-hand term by its lower bound and recursively until we have subsets of two variables:

$$\geq \arg\max_S \sum_{i\in S}\sum_{j\in S} I(X_{S-(i,j)};Y) \geq \arg\max_S \sum_{i\in S}\sum_{j\in S} I(X_{i,j};Y) \qquad (14)$$

In other words, without any information on how to combine subsets made of more than two variables, the most promising subset (the best bound) is the one with the highest sum of mutual information on all the combinations of two variables. We choose to stop the recursivity at two variables because it is the minimal size of subset that can capture variable complementarity (i.e. $I(X_{i,j};Y) = I(X_i;Y) + I(X_j;Y) + C_Y(X_i;X_j)$). Note that this strategy, when we stop the recursion at one variable, boils down to the ranking approach.

A similar approach has been developped in [12] based on the Möbius representation of mutual information. However, in order to improve the selection procedure we use here a normalized measure of mutual information very close to the symmetrical uncertainty presented in [9]: the *symmetrical relevance.*

Definition 5. *Given two random variables X,Y a joint probability distribution $p(x,y)$, the symmetrical relevance $SR(X,Y)$ is defined as:*

$$SR(X;Y) = \frac{I(X,Y)}{H(X,Y)} \qquad (15)$$

This definition expresses that symmetrical relevance is a function $0 \leq SR(X;Y) \leq 1$ that indicates the "concentration" of mutual information "contained" in $p(x,y)$.

As a consequence, our resulting criterion is the following:

$$X_{DISR} = \arg\max_{X_i\in X_{-S}} \{\sum_{X_j\in X_S} SR(X_{i,j};Y)\} \qquad (16)$$

The main advantage in using this criterion for selecting variables is that a complementary variable of an already selected one has a much higher probability to be selected than with other criteria. As this criterion measures symmetrical relevance on all the combination of two variables (double input) of a subset, we have called the criterion: the *double input symmetrical relevance* (DISR). At the

Table 1. Qualitative comparison of different information theoretic filters, according to different aspects: relevance selection, redundancy avoidance, complementarity selection and multivariate densities avoidance

criterion	rank	REL	MRMR	CMIM	DISR
relevance selection	V	V	V	V	V
redundancy avoidance	–	V	V	V	V
complementarity selection	–	V	–	–	V
multivariate density avoidance	V	–	V	V	V

dth step of the forward search, the search algorithm computes $n - d$ evaluations where each evaluation requires the estimation of d tri-variate densities (one for each previously selected variable). In the next section, the DISR criterion is assessed and compared with the other heuristic search filters discussed in the Section 3.

Table 1 summarizes the methods discussed so far in terms of some peculiar aspects: the capacity of selecting relevant variables, of avoiding redundancy, of selecting complementary features and of avoiding the computation of multivariate densities.

5 Experiments

A major topic in bioinformatics is how to build accurate classifiers for cancer diagnostic and prognostic purposes on the basis of microarray genomic signatures. This task can be considered as a challenging benchmark for feature selection algorithms [7] given the high feature to sample ratio.

We use eleven public domain multi-class datasets from [14] (Table 2) in order to assess and compare our technique with the state-of-the-art approaches.

In our experimental framework, each continuous variable has been discretized in equal sized interval. The number of intervals of each input is based on the Scott criterion, see [15]. All the datasets are partitioned into two parts: a selection set and a test set (each having size equal to $N/2$). We compare the filter based on DISR with the four state-of the art approaches discussed above: a Ranking algorithm and three filters based on the Relevance criterion, the Minimum Redudancy Maximum Relevance criterion and the Conditional Mutual Information

Table 2. The 11 datasets of microarray cancer from `http://www.tech.plym.ac.uk`. The column n represents the number of probes in the microarray, the column N the number of samples and the column c the number of classes. The remaining columns contain the average accuracy of each selection method averaged over the three classifiers (SVM, 3-NN, naive Bayes). The accuracy of the two best methods for each dataset is typed in bold face.

Dataset (DN)	n	N	c	Rank	REL	CMIM	MRMR	DISR
11_Tumors	12534	87	11	**49%**	46%	48%	42%	**49%**
14_Tumors	15010	308	26	22%	**25%**	20%	**26%**	19%
9_Tumors	5727	60	9	19%	**36%**	20%	23%	**28%**
Leukemia1	5328	72	3	71%	**74%**	71%	69%	**78%**
Leukemia2	11226	72	3	**68%**	57%	62%	65%	**67%**
Prostate_Tumor	10510	102	2	**76%**	66%	62%	**78%**	73%
Brain_Tumor1	5921	90	5	**73%**	70%	70%	70%	**71%**
Brain_Tumor2	10368	50	4	47%	47%	49%	**59%**	**60%**
Lung_Cancer	12601	203	5	**84%**	74%	**82%**	77%	74%
SRBCT	2309	83	4	70%	53%	**75%**	**75%**	67%
DLBCL	5470	77	2	77%	**88%**	70%	71%	**88%**

Table 3. Statistically (0.1 level and 0.2 level of a paired two-tailed t-test) significant wins, ties or losses over best first search combined with DISR criterion

W/T/L VS DISR	Rank	REL	CMIM	MRMR	Rank	REL	CMIM	MRMR
3-NN	0/9/2	1/8/2	1/7/3	2/7/2	1/5/5	1/7/3	1/7/3	2/7/2
Naive Bayes	3/8/0	2/7/2	1/8/2	1/9/1	4/7/0	3/6/2	1/8/2	1/9/1
SVM	2/9/0	2/6/3	2/6/3	2/8/1	2/6/3	3/4/4	2/5/4	3/5/3

Maximization criterion, respectively. Each selection method stops after that 15 variables have been selected. Then, the evaluation of the selection is done on the test set, by using a ten-fold cross validation with a 3-nearest neighbor, a naive Bayes and a SVM learning algorithm with a radial kernel. Each learning technique led to the choice of a different number of variables in a range from 2 to 15. Then for each of the eleven datasets and for each selection method, the best number of variables and the classification accuracy is computed. A set of statistical paired t-test on the set of classification errors are reported in Table 3.

As far as the implementation of the three learning methods is concerned, we used the algorithms made available by the R statistical language [16].

According to Table 2, the DISR criterion outperforms slightely all the other methods in terms of average accuracy. Furthermore, our method is one of the two best methods for 7 out of 11 datasets.

Table 3 reports the significant wins, ties or losses (at 0.1 and 0.2 significance levels of a paired two-tailed t-test, respectively) of the DISR criterion against all the other. We remark that in the case a 3-Nearest Neighbor, the DISR criterion is equivalent to MRMR and better than all the other methods. For a naive Bayes classifier, the performances of the DISR are slightly lower. This is not surprising because the benefits of the DISR criterion are related to variable complementarity whereas the success of a naive Bayes classifier typically relies on the opposite, that is variable independence. As far as the SVM classifier is concerned, at the 0.1 significance level, DISR appears to be slightly better than both REL and CMIM, and slightly worse than RANK and MRMR. However, at 0.2 significance level the DISR outperforms all the other methods except MRMR.

6 Conclusion and Future Work

This paper formalized an original notion in feature selection: variable complementarity. Also, a lower bound on the mutual information of a subset of variables with the output was demonstrated. On the basis of these considerations, we proposed a new selection criterion: the double input symmetrical relevance (DISR). The experimental session shows that this criterion is promising in high feature-to-sample ratio classifaction tasks like gene expression microarray datasets. Note that in gene selection, variable complementarity can be biologically meaningful since it is common to observe combination of genes acting together.

Further experiments will focus on (i) datasets with more samples and/or less features, (ii) other search strategies than the forward selection in order to validate the criterion in a wider range of domains, (iii) the impact of the discretization method to the efficiency of the feature selection algorithms.

References

1. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182
2. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97** (1997) 273–324
3. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence **97** (1997) 245–271
4. Provan, G., Singh, M.: Learning bayesian networks using feature selection. In: in Fifth International Workshop on Artificial Intelligence and Statistics. (1995) 450–456
5. Duch, W., Winiarski, T., Biesiada, J., Kachel, A.: Feature selection and ranking filters. In: International Conference on Artificial Neural Networks (ICANN) and International Conference on Neural Information Processing (ICONIP). (2003) 251–254
6. Bell, D.A., Wang, H.: A formalism for relevance and its application in feature subset selection. Machine Learning **41** (2000) 175–195
7. Peng, H., Long, F.: An efficient max-dependency algorithm for gene selection. In: 36th Symposium on the Interface: Computational Biology and Bioinformatics. (2004)
8. Fleuret, F.: Fast binary feature selection with conditional mutual information. Journal of Machine Learning Research **5** (2004) 1531–1555
9. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research **5** (2004) 1205–1224
10. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley, New York (1990)
11. Yang, H., Moody, J.: Feature selection based on joint mutual information. In: In Advances in Intelligent Data Analysis (AIDA), Computational Intelligence Methods and Applications (CIMA), Rochester New York, ICSC (1999)
12. Kojadinovic, I.: Relevance measures for subset variable selection in regression problems based on k-additive mutual information. Computational Statistics and Data Analysis **49** (2005) 1205–1227
13. Meyer, P.: Information theoretic filters for feature selection. Technical report, Universite Libre de Bruxelles ((548) 2005)
14. web: (http://www.tech.plym.ac.uk/spmc/bioinformatics/microarray_cancers.html)
15. Scott, D.W.: Multivariate Density Estimation. Theory,. Wiley (1992)
16. R-project: (www.r-project.org)

Comparison of Neural Network Optimization Approaches for Studies of Human Genetics

Alison A. Motsinger, Scott M. Dudek, Lance W. Hahn, and Marylyn D. Ritchie

Center for Human Genetics Research, Department of Molecular Physiology & Biophysics, Vanderbilt University, Nashville, TN, 37232, USA
{motsinger, dudek, hahn, ritchie}@chgr.mc.vanderbilt.edu
http://chgr.mc.vanderbilt.edu/ritchielab

Abstract. A major goal of human genetics is the identification of susceptibility genes associated with common, complex diseases. The preponderance of gene-gene and gene-environment interactions comprising the genetic architecture of common diseases presents a difficult challenge. To address this, novel computational approaches have been applied to studies of human disease. These novel approaches seek to capture the complexity inherent in common diseases. Previously, we developed a genetic programming neural network (GPNN) to optimize network architecture for the detection of disease susceptibility genes in association studies. While GPNN was a successful endeavor, we wanted to address the limitations in its flexibility and ease of development. To this end, we developed a grammatical evolution neural network (GENN) approach that accounts for the drawbacks of GPNN. In this study we show that this new method has high power to detect gene-gene interactions in simulated data. We also compare the performance of GENN to GPNN, a traditional back-propagation neural network (BPNN) and a random search algorithm. GENN outperforms both BPNN and the random search, and performs at least as well as GPNN. This study demonstrates the utility of using GE to evolve NN in studies of complex human disease.

1 Introduction

The identification and characterization of susceptibility genes for common complex human diseases, such as hypertension, is a difficult challenge[1,2]. This is largely due to the complexity of these diseases, and the likelihood that many disease susceptibility genes exhibit effects that are dependent partially or solely on interactions with other genes and the environment. These interactions, known as epistasis, are difficult to detect using traditional statistical methods[3]. Thus, a number of novel statistical and computational methods have been developed[4-11]. Neural networks (NN) are a supervised pattern recognition method commonly used in many fields for data mining. The back propagation NN (BPNN) is one of the most commonly used NN[12] and is the NN chosen for most genetic epidemiology studies [13-21]. Successful use of NN for data mining requires defining an optimal NN architecture for the problem at hand. However, it is not always intuitive what the optimal architecture should be for a given dataset and, as a result, a cumbersome trial and error approach is often taken.

Previously, we implemented a neural network optimized via genetic programming (GPNN)[22]. Optimizing neural network architecture with genetic programming was first proposed by Koza and Rice[23]. We implemented and extended the GPNN approach for use in association studies of human disease. The goal of GPNN was to improve upon the trial-and-error process of choosing an optimal architecture for a pure feed-forward back propagation neural network[22]. GPNN optimizes the inputs from a large pool of variables, the weights, and the connectivity of the network - including the number of hidden layers and the number of nodes in the hidden layer. Thus, the algorithm automatically generates optimal neural network architecture for a given dataset. This gives it an advantage over the traditional back propagation NN, in which the inputs and architecture are pre-specified and only the weights are optimized.

GPNN was a successful endeavor – it has shown high power to detect gene-gene interactions in both simulated and real data[24]. Still, there are limitations to evolving NN using this type of machine learning algorithm. First, the GP implementation that was used for GPNN involves building binary expression trees. Therefore, each node is connected to exactly two nodes at the level below it in the network. This did not seem to hinder the power of GPNN in smaller datasets[22,24-26]; however, we hypothesize that for more complex data, more complicated NN will be required, and two connections per node may not be sufficient. Second, changes to GPNN require altering and recompiling source code, which hinders flexibility and increases development time. For example, GPNN is limited in the depth of the network. This means there is a limit to the number of levels the network can contain. Again, this was not a hindrance for GPNN in the previous power studies[22,24-26], but this may not scale well for more complex datasets.

In response to these concerns, we developed a NN approach for detecting gene-gene interactions that uses grammatical evolution (GE) as a strategy for the optimization of the NN architecture. Grammatical evolution (GE) is a variation on genetic programming that addresses some of the drawbacks of GP[27,28]. GE has been shown to be effective in evolving Petri Nets, which are discrete dynamical systems that look structurally similar to neural networks, used to model biochemical systems[29]. By using a grammar, substantial changes can be made to the way that NN are constructed through simple manipulations to the text file where the grammar is specified. No changes in source code are required and thus, there is no recompiling. The end result is a decrease in development time and an increase in flexibility. These two features are important improvements over GPNN.

Preliminary studies with GPNN show that an evolutionary optimization is more powerful than traditional approaches for detecting gene-gene interactions. We have shown that the GPNN strategy is able to model and detect gene-gene interactions in the absence of main effects in many epistasis models with higher power than back propagation NN[22], stepwise logistic regression[26], and a stand alone GP[25]. GPNN has also detected interactions in a real data analysis of Parkinson's disease[24]. Similarly to GPNN, the grammatical evolution optimized neural network (GENN) optimizes the inputs from a pool of variables, the synaptic weights, and the architecture of the network. The algorithm automatically selects the appropriate network architecture for a particular dataset. Thus, if GE allows for more flexible evolution of NN, can it perform as well or better than GPNN?

Because GENN retains the beneficial properties of GPNN and offers substantial improvements in terms of flexibility and development, we hypothesize that NN optimized by GE will exhibit power equal to and even exceeding GPNN. In this study, we compare the performance of GENN to a more traditional back-propagation NN, a random search algorithm, and GPNN. We find that both GENN and GPNN outperform the random search and the BPNN. We also show that GENN is equal in power to GPNN in detecting gene-gene interactions in our simulated disease models.

2 Methods

2.1 Grammatical Evolution Neural Network (GENN)

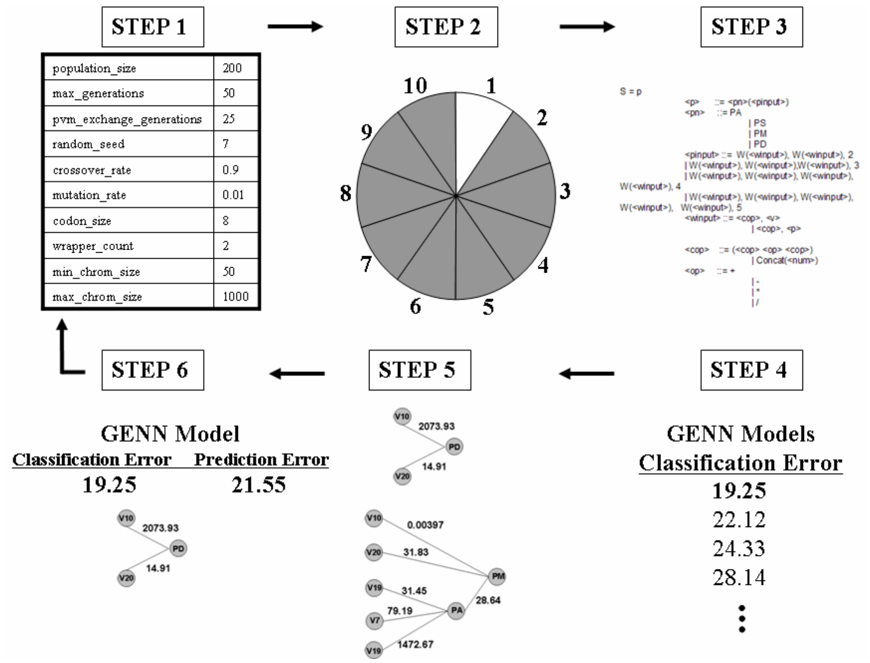
Grammatical Evolution (GE) is a form of evolutionary computation that allows the generation of computer programs using grammars[27,28]. GE uses populations made of linear genomes that are translated by the grammar. Each individual consists of a binary genome divided into codons. Mutation can occur on individual bits along the genome, but crossover only occurs between codons. These codons are translated according to the grammar into a resulting phenotype (in this case, a functional NN). The resulting individual/phenotype can be tested for fitness and evolutionary operators are applied to create subsequent generations. By using the grammar to map a NN, GE separates genotype from phenotype. This allows for greater genetic diversity within a population than offered by other evolutionary algorithms, like GP. Since GENN uses a grammar to define the structure of the resulting NN, we can easily vary the behavior of the program with changes to the grammar.

GE differs from GP in several ways. First, unlike GP, GE uses a linear genome - similar to a genetic algorithm. Second, GE performs mapping from genotype to phenotype using the rules of a grammar, much like the "rules" of the biological process of DNA transcription into mRNA. Finally, all evolutionary processes take place at the chromosomal level (binary strings) rather than the phenotypic level (binary expression tree). Ultimately, the goal of GP and GE is synonymous: to evolve computer programs using evolutionary processes[27,28].

A detailed description of GE can be found in O'Neill and Ryan[28]. Briefly, a Backus-Naur Form (BNF) grammar must be defined for the process of genotype-to-phenotype mapping. A variable-length binary string genome is used in a genetic algorithm, with a set of 8 bits constituting a codon. Each binary codon represents an integer value used to select a rule in the grammar. One non-terminal element is designated by the grammar as the start element. The mapping process proceeds as the first codon maps the start symbol of the solution to a rule by generating an integer value from the 8 bits. To select a rule, the operator used is {(codon integer value) MOD (number of rules)}. The start element is replaced by the elements of the rule selected and this process proceeds until only terminal elements remain. A wrapping process can be used if the program has non-terminals at the point at which the end of the chromosome has been reached so that the algorithm returns to the start of the chromosome to obtain the next codon. This wrapping process can be allowed to occur *N* times as necessary, where *N* is defined in the configuration file. The resulting phenotype is a NN, which can then be evaluated for fitness.

The steps of GENN are similar to GPNN[22]. First, GENN has a set of parameters that must be initialized in the configuration file. Second, the data are divided into 10 equal parts for 10-fold cross-validation. Here, we train GENN on 9/10 of the data to develop a NN model. Later, we test this model on the other 1/10 of the data to evaluate the predictive ability of the model. This approach has been described in detail for GPNN[22]. Third, training of GENN begins by generating an initial population of random solutions. Each solution is generated via sensible initialization[28]. Using sensible initialization, an initial population is generated that creates functioning NN. In the sensible initialization step an expression tree is created using the grammar. The software assigns a minimum depth to each rule that describes the depth required for the rule to be completed. As each tree is built, the algorithm randomly selects only rules that can fit within the remaining depth of the tree. Half of the individual NN are built to the maximum depth by only selecting recursive rules until a non-recursive rule must be chosen to complete the tree and half are generated to a random depth no greater than the maximum by selecting any rule that can fit in the remaining depth of the tree. The final step in initialization is to convert nodes of the tree into corresponding codons. Fourth, each NN is evaluated on the training set and its fitness recorded. Fifth, the best solutions are selected for crossover and reproduction using a selection technique. The selection method can be specified in the configuration file, where the options are uniform, rank, roulette, and tournament[30]. A proportion of the best solutions will be directly copied (reproduced) into the new generation. Another proportion of solutions will be used for crossover with other best solutions. The crossover is performed at the chromosomal level, not at the level of the expression tree. The new generation, which is equal in size to the original population, begins the cycle again. This continues until some criterion is met, after which GENN stops. This criterion is either a classification error of zero or a limit on the number of generations. An optimal solution is identified after each generation. At the end of GENN evolution, the overall best solution is selected as the optimal NN. Sixth, this best GENN model is tested on the 1/10 of the data left out to estimate the prediction error of the model. Steps two through six are performed ten times with the same parameters settings, each time using a different 9/10 of the data for training and 1/10 of the data for testing. An overview of the GENN algorithm is shown in Figure 1.

We have implemented GE to optimize inputs, architecture, and weights of a NN. The grammar used is available from the authors upon request. The GA used to evolve the binary string that is transcribed into a NN has the following parameters in the current implementation: crossover rate = 0.9, mutation = 0.01, population = 200, max generations = 50, codon size = 8, GE wrapping count = 2, min chromosome size (in terms of codons) = 50, max chromosome size = 1000, selection = roulette (can also be uniform, rank, or tournament), and sensible initialization depth = 10. To prevent stalling in local minima, the island model of parallelization is used, where the best individual is passed to each of the other processes after every 25 generations[31]. The genome is derived from GAlib (version 2.4.5) which is freely available at http://lancet.mit.edu/ga/dist/, and a typical GA one-point crossover of linear chromosomes is used.



population_size	200
max_generations	50
pvm_exchange_generations	25
random_seed	7
crossover_rate	0.9
mutation_rate	0.01
codon_size	8
wrapper_count	2
min_chrom_size	50
max_chrom_size	1000

```
S = p
          <p>      ::= <pn>(<pinput>)
          <pn>     ::= PA
                       | PS
                       | PM
                       | PD
          <pinput> ::=  W(<winput>), W(<winput>), 2
          | W(<winput>), W(<winput>),W(<winput>), 3
          | W(<winput>), W(<winput>), W(<winput>),
W(<winput>), 4
          | W(<winput>), W(<winput>), W(<winput>),
W(<winput>),   W(<winput>), 5
          <winput> ::= <cop>, <v>
                       | <cop>, <p>

          <cop>    ::= (<cop> <op> <cop>)
                       | Concat(<num>)
          <op>     ::= +
                       | -
                       | *
                       | /
```

Fig. 1. An overview of the GENN method. The steps correspond to the description of the method in Section 2.1.

Classification error is calculated on the set of training data as the fitness metric. As mentioned earlier, the dataset is divided into cross-validation subsets. GENN is optimized using a training set of data, and a subset of the data is left out as a test set to evaluate the final solution and prevent over-fitting. Classification error refers to the number of samples in the training dataset that are incorrectly classified by the network. Prediction error, which refers to the number of samples in the test dataset that are incorrectly classified using the GENN model generated during training, is used for final model selection. The overall goal of the learning process is to find genetic models that accurately classify the data. Cross-validation is used in conjunction with this learning process to produce a model that not only can accurately classify the data at hand, but can predict on future, unseen data.

2.2 Genetic Programming Neural Networks (GPNN)

GPNN uses genetic programming to determine the optimal architecture for neural networks. Both the method and the software have previously been described[22]. GPNN was applied as presented in the references. Like GENN, models are trained on classification error, and a cross validation consistency and prediction error are determined for the final model. Unlike GENN, previous studies have shown cross validation consistency as the best criterion for final model selection. However for this study, the results are identical whether you use cross-validation consistency or prediction error for final model selection. All configuration parameters are identical to those in GENN. This will allow for a direct comparison of GPNN and GENN.

2.3 Random Search Algorithm

As a negative control, a random-search algorithm was implemented. The random search algorithm uses the same fitness metric as both GENN and GPNN. The random search generates the initial chromosome population as described above for GENN, using sensible initialization, but this new random population occurs at every generation instead of only at the beginning of the run. Genotype-to-phenotype mapping is performed just as it is for GENN. The algorithm stores the single best network over all generations and returns that as the final model. All other networks are discarded.

2.4 Back Propagation Neural Network

In this study, we used a traditional fully-connected, feed-forward network comprised of one input layer, zero, one or two hidden layers, and one output layer, trained by back-propagation. The software used, the NICO toolkit, was developed at the Royal Institute of Technology, (http://www.speech.kth.se/NICO/index.html).

Defining the network architecture is an important decision that can greatly affect the results of the analysis[32]. There are several strategies utilized for architecture selection, including a prediction error fitness measure such that an architecture is selected by its generalization to new observations[32], or a classification (training) error metric[33]. Because we use cross-validation to verify the generalizability of our models, and to make a more fair comparison to the other methods in this study, we used classification error as a basis for evaluating and making changes to the BPNN architecture. We began with a very small network, and several parameters were varied to obtain an appropriate architecture for each dataset, including: the number of hidden layers, the number of nodes in the hidden layer, and the learning momentum (the fraction of the previous change in a weight that is added to the next change). This trial and error approach is typically employed for optimization of BPNN architecture[33,34]. BPNN was implemented as described in [22]. Final model selection was performed based on lowest prediction error, as with GENN.

2.5 Data Simulation

Epistasis, or gene-gene interaction, occurs when the phenotype under study cannot be predicted from the independent effects of any single gene, but is the result of combined effects of two or more genes[34]. It is increasingly accepted that epistasis plays an important role in the genetic architecture of common genetic diseases[35]. Penetrance functions are used to represent epistatic genetic models in this simulation study. Penetrance defines the probability of disease given a particular genotype combination by modeling the relationship between genetic variations and disease risk.

For our power studies, we simulated case-control data using two different epistasis models exhibiting interaction effects in the absence of main effects. Models that lack main effects are desirable because they challenge the method to find gene-gene interactions in a complex dataset. Also, a method able to detect purely interactive terms will be likely to identify main effects as well.

Table 1. Multilocus penetrance functions used to simulate case-control data exhibiting gene-gene interactions in the absence of main effects. Penetrance is calculated as p(disease|genotype). Marginal penetrance values (not shown) are all equal for each model.

a. Model 1

	BB	Bb	bb
AA	0	.10	0
Aa	.10	0	.10
aa	0	.10	0

b. Model 2

	BB	Bb	bb
AA	0	0	.10
Aa	0	.50	0
Aa	.10	0	0

To evaluate the power of the above methods for detecting gene-gene interactions, we simulated case-control data using two different two-locus epistasis models in which the functional loci are single nucleotide polymorphisms (SNPs). The first model was initially described by Li and Reich[36], and later by Moore [37]. This model is based on the nonlinear XOR function[38] that generates an interaction effect in which high risk of disease is dependent on inheriting a heterozygous genotype (Aa) from one locus or a heterozygous genotype (Bb) from a second locus, but not both. The high risk genotypes are AaBB, Aabb, AABb, and aaBb, all with penetrance of 0.1 (Table 1a). The proportion of the trait variance that is due to genetics, or heritability, of this model is low. Specifically, as calculated according to Culverhouse *et al*[39], the heritability is 0.053. The second model was initially described by Frankel and Schork[40], and later by Moore[37]. In this second model, high risk of disease is dependent on inheriting exactly two high risk alleles (A and/or B) from two different loci. In this model, the high risk genotypes were AAbb, AaBb, and aaBB, with penetrance of 0.1, 0.5, and 0.1 respectively (Table 1b). The heritability of this model is 0.051.

These models were selected because they exhibit interaction effects in the absence of any main effects when genotypes were generated according to Hardy-Weinberg proportions (in both models, p=q=0.5). For both models, we simulated 100 datasets consisting of 200 cases and 200 controls, each with 10 SNPs, 2 of which were functional. The data were generated using the software package described by Moore *et al*[37]. Dummy variable encoding was used for each dataset, where *n-1* dummy variables were used for *n* levels[19]. Data were formatted with rows representing individuals and columns representing dummy-encoded genotypes with the final column representing disease status. Though biological relevance of these models is uncertain, they do represent a “worst case scenario” in the detection of epistasis. If a method performs well under such minimal effects, it is predicted it will also perform well in identifying gene-gene interactions in models with greater effect sizes.

2.6 Data Analysis

We used all four methods (GENN, GPNN, BPNN and random search) to analyze both epistasis models. The configuration parameter settings were identical for GENN, GPNN and the random search (without evolutionary operators for random search): 10 demes, migration every 25 generations, population size of 200 per deme, 50 generations, crossover rate of 0.9, and a reproduction rate of 0.1. For GENN and the random search, prediction error was used for final model selection as described in Section 2.1. For GPNN, cross-validation consistency was calculated for each model and the final model was selected based on this metric (as described in [22]). Sensible initialization was used in all three algorithms.

For our traditional BPNN analysis, all possible inputs were used and the significance of each input was calculated from its input relevance R_I, where R_I is the sum of squared weights for the ith input divided by the sum of squared weights for all inputs[38]. Next, we performed 1000 permutations of the data to determine what input relevance was required to consider a SNP significant in the BPNN model (data not shown). This empirical range of critical relevance values for determining significance was 10.43% - 11.83% based on the permutation testing experiments. Cross validation consistency was also calculated and an empirical cutoff for the cross validation consistency was determined through permutation testing (using 1000 randomized datasets). This cutoff was used to select SNPs that were functional in the epistasis model for each dataset. A cross validation consistency of greater than 5 was required to be statistically significant.

Power for all analyses is reported under each epistatic model as the number of times the algorithm correctly identified the correct functional loci (both with and without any false positive loci) over 100 datasets. Final model selection was performed for each method based on optimum performance in previous studies[22]. If either one or both of the dummy variables representing a single SNP was selected, that locus was considered present in the model.

3 Results

Table 2 lists the power results from all four algorithms. Because of the small size of the dataset, all four algorithms performed reasonably well. With a limited number of SNPs, these learning algorithms can effectively become exhaustive searches. As hypothesized, GENN and GPNN both out-performed the traditional BPNN and the random search. The performance of GENN and GPNN were consistent, as expected. This demonstrates that GENN will work at least as well as GPNN, while allowing for faster development and more flexible use. Because the number of variables included in the dataset was small, the random search performed reasonably well, as the trial and error approach had a limited number of variables to search through. As Table 2 shows, there is a large gap in the performance of the random search between Model 1 and Model 2. This is probably due to the difference in the difficulty inherent in the two models. The power of BPNN to detect Model 2 was also lower than for Model 1, indicating a difference in the challenge of modeling the different models. Additionally, the stochastic nature of a random algorithm can lead to erratic results, as shown here. These erratic power results further demonstrate the utility of an evolutionary approach to optimizing NN architecture. The random search even outperformed BPNN for Model 1, probably because the random search was able to search through more possible NN architectures than BPNN so was able to find the correct model more often in these simulations.

Table 3 summarizes the average classification error (training error) and prediction error (testing error) for the four algorithms evaluated using the 100 datasets for each model. Due to the probabilistic nature of the functions used in the data simulation,

Table 2. Power (%) for each method on both gene-gene interaction models (with no false positive loci)

Epistasis Model	GENN	GPNN	BPNN	Random Search
1	100	100	53	87
2	100	100	42	10

Table 3. Results for all four algorithms, demonstrating average classification error (CE) and prediction error (PE) for each epistasis model. The range of observed observations is listed below the average.

Model	GENN		GPNN		BPNN		Random Search	
	CE	PE	CE	PE	CE	PE	CE	PE
1	0.237 (.212-.254)	0.237 (.210-.255)	0.237 (.200-.284)	0.237 (.208-.279)	0.008 (.000-.183)	0.340 (.201-.410)	0.236 (.088-.483)	0.237 (.075-.400)
2	0.243 (.212-.260)	0.243 (.209-.271)	0.242 (.212-.261)	0.243 (.210-.269)	0.008 (.000-.181)	0.303 (.240-.410)	0.242 (.080-.494)	0.245 (.075-.400)

Table 4. Power (%) for each method to detect functional SNPs in both gene-gene interaction models (with or without false positive loci)

Model	GENN		GPNN		BPNN		Random Search	
	SNP 1	SNP 2	SNP 1	SNP 2	SNP 1	SNP 2	SNP 1	SNP 2
1	100	100	100	100	88	90	100	100
2	100	100	100	100	80	82	100	100

there is some degree of noise present in the data. The average error inherent in the 100 Model 1 datasets is 24%, and the error in the Model 2 datasets is 18%. As the table shows, GENN, GPNN and the random search all had error rates closely reflecting the real amount of noise in the data. Those three algorithms had lower prediction errors than BPNN, while BPNN had lower classification errors for both models. The lower classification error is due to model over-fitting. The other three algorithms, including even the random search, are better able to model gene-gene interaction and develop NN models that can generalize to unseen data. While the random search did not demonstrate the same degree of over-fitting experienced with BPNN, the averages reported here disguise the fact that the range of errors across datasets was very high. While the average errors for the random search look similar to those for GPNN and GENN, the range of observed values was much larger, implying that the random search also tends to over-fit. We speculate that GENN and GPNN are not over-fitting because, while these methods are theoretically able to build a tree with all variables included, neither method is building a fully connected NN using all variables.

To further understand the behavior of the algorithms, and in particular the seemingly inconsistent results of the random search's average errors and low relative power, we calculated power for each functional locus as the proportion of times a SNP was included in the final model (regardless of what other SNPs are present in the model) for all datasets. Table 4 lists the results of this power calculation for all four methods. The tendency of the random search algorithm to over-fit models becomes clear in the comparisons of Tables 2-4. The random search finds the functional SNPs, but includes many false positive loci, which is highly undesirable since the end goal of association analysis is variable selection. The same false positive trend holds true for BPNN.

4 Discussion

We have demonstrated that grammatical evolution is a valid approach for optimizing the architecture of NN. We have shown that GENN outperforms both a random search and traditional BPNN for analysis of simulated epistasis genetic models. Because of the small number of SNPs in this study, both BPNN and the random search NN had modest power, as one would expect. With a small number of variables to test, examining low-order combinations is relatively easy. As the number of variables increases, the resultant combinatorial explosion limits the feasibility of trial and error approaches. Moody[33] demonstrates that enumeration of all possible NN architectures is impossible, and there is no way to know if a globally optimal architecture is selected. The performance gap between the evolutionarily optimized algorithms and the trial and error approaches is expected to widen as the number of variables increases.

Additionally, we show that GENN performs at least as well as GPNN. Because of the limited number of noise variables, and the fact that these two methods reached the upper limit of power, a more extensive comparison between GENN and GPNN needs to be performed. Power will need to be studied in a range of datasets, demonstrating a wide range of heritability values and number of noise variables. Because of the greater flexibility of GE compared to GP, we predict that GENN will out-perform GPNN on more complex datasets.

Because the end-goal of these methods is variable selection, performance has been evaluated according to this metric in this study. In future studies, it would be interesting to evaluate the architectures of the NN that are constructed by these different methods to further evaluate the differences in their performance. Other measures of model fitness, such as sensitivity and specificity could also be dissected in evaluating the performance of GENN.

Also, while simulated data are necessary in method development, the eventual purpose of this method is for the analysis of real data. GENN will need to be tested on real case-control genetic data.

This study introduces a novel computational method and demonstrates that GENN has the potential to mature into a useful software tool for the analysis of gene-gene interactions associated with complex clinical endpoints. The ease of flexibility and ease of development of utilizing a grammar will aid in additional studies with this method.

Acknowledgements

This work was supported by National Institutes of Health grants HL65962, GM62758, and AG20135. We would also like to thank David Reif for his helpful comments on the manuscript.

References

1. Kardia S, Rozek L, Hahn L, Fingerlin T, Moore J: Identifying multilocus genetic risk profiles: a comparison of the multifactor data reduction method and logistic regression. *Genetic Epidemiology* 2000.
2. Moore JH, Williams SM: New strategies for identifying gene-gene interactions in hypertension. *Ann Med* 2002, 34: 88-95.
3. Culverhouse R, Klein T, Shannon W: Detecting epistatic interactions contributing to quantitative traits**.** *Genet Epidemiol* 2004, 27: 141-152.
4. Hahn LW, Ritchie MD, Moore JH: Multifactor dimensionality reduction software for detecting gene-gene and gene-environment interactions. *Bioinformatics* 2003, 19: 376-382.
5. Kooperberg C, Ruczinski I, LeBlanc ML, Hsu L: Sequence analysis using logic regression. *Genet Epidemiol* 2001, 21 Suppl 1: S626-S631.
6. Moore JH: The ubiquitous nature of epistasis in determining susceptibility to common human diseases. *Hum Hered* 2003, 56: 73-82.
7. Nelson MR, Kardia SL, Ferrell RE, Sing CF: A combinatorial partitioning method to identify multilocus genotypic partitions that predict quantitative trait variation. *Genome Res* 2001, 11: 458-470.
8. Ritchie MD, Hahn LW, Roodi N, Bailey LR, Dupont WD, Parl FF *et al.*: Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer**.** *Am J Hum Genet* 2001, 69: 138-147.
9. Ritchie MD, Hahn LW, Moore JH: Power of multifactor dimensionality reduction for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. *Genet Epidemiol* 2003, 24: 150-157.
10. Tahri-Daizadeh N, Tregouet DA, Nicaud V, Manuel N, Cambien F, Tiret L: Automated detection of informative combined effects in genetic association studies of complex traits. *Genome Res* 2003, 13: 1952-1960.
11. Zhu J, Hastie T: Classification of gene microarrays by penalized logistic regression**.** *Biostatistics* 2004, 5: 427-443.
12. Schalkoff R: *Artificial Neural Networks*. New York: McGraw-Hill Companies Inc.; 1997.
13. Bhat A, Lucek PR, Ott J: Analysis of complex traits using neural networks. *Genet Epidemiol* 1999, 17: S503-S507.
14. Curtis D, North BV, Sham PC. Use of an artificial neural network to detect association between a disease and multiple marker genotypes. Annals of Human Genetics 65, 95-107. 2001.
15. Li W, Haghighi F, Falk C: Design of artificial neural network and its applications to the analysis of alcoholism data. *Genet Epidemiol* 1999, 17: S223-S228.
16. Lucek P, Hanke J, Reich J, Solla SA, Ott J: Multi-locus nonparametric linkage analysis of complex trait loci with neural networks. *Hum Hered* 1998, 48: 275-284.
17. Lucek PR, Ott J: Neural network analysis of complex traits. *Genet Epidemiol* 1997, 14: 1101-1106.
18. Marinov M, Weeks D: The complexity of linkage analysis with neural networks. *Human Heredity* 2001, 51: 169-176.
19. Ott J. Neural networks and disease association. American Journal of Medical Genetics (Neuropsychiatric Genetics) 105[60], 61. 2001.
20. Saccone NL, Downey TJ, Jr., Meyer DJ, Neuman RJ, Rice JP: Mapping genotype to phenotype for linkage analysis. *Genet Epidemiol* 1999, 17 Suppl 1: S703-S708.
21. Sherriff A, Ott J: Applications of neural networks for geen finding. *Advances in Genetics* 2001, 42: 287-297.

22. Ritchie MD, White BC, Parker JS, Hahn LW, Moore JH: Optimization of neural network architecture using genetic programming improves detection and modeling of gene-gene interactions in studies of human diseases. *BMC Bioinformatics* 2003, 4: 28.
23. Koza J, Rice J: Genetic generation of both the weights and architecture for a neural network. *IEEE Transactions* 1991, II.
24. Motsinger AA, Lee S, Mellick G, Ritchie MD: GPNN: Power studies and applications of a neural network method for detecting gene-gene interactions in studies of human disease. *BMC Bioinformatics* 2005, in press.
25. Bush WS, Motsinger AA, Dudek SM, Ritchie MD: Can neural network constraints in GP provide power to detect genes associated with human disease? *Lecture Notes in Computer Science* 2005, 3449: 44-53.
26. Ritchie MD, Coffey CSMJH: Genetic programming neural networks: A bioinformatics tool for human genetics. *Lecture Notes in Computer Science* 2004, 3102: 438-448.
27. O'Neill M, Ryan C. Grammatical Evolution. IEEE Transactions on Evolutionary Computation 5, 349-357. 2001.
28. O'Neill M, Ryan C. Grammatical evolution: Evolutionary automatic programming in an arbitrary language. 2003. Boston, Kluwer Academic Publishers.
29. Moore JH, Hahn LW: Petri net modeling of high-order genetic systems using grammatical evolution. *BioSystems* 2003, 72: 177-186.
30. Mitchell M. An introduction to genetic algorithms. 1996. Cambridge, MIT Press.
31. Cantu-Paz E. Efficient and accurate parallel genetic algorithms. 2000. Boston, Kluwer Academic Publishers.
32. Utans J, Moody J. Selecting neural network architectures via the prediction risk application to corporate bond rating prediction. 1991. Los Alamitos, California, IEEE Press. Conference Proceedings on the First International Conference on Artificial Intelligence Applications on Wall Street.
33. Moody J: Prediction risk and architecture selection for neural networks. In *From Statistics to Nerual Networks: Theory and Pattern Recognition Applications*. Edited by Cherkassky V, Friedman JH, Wechsler H. NATO ASI Series F, Springer-Verlag; 1994.
34. Fahlman SE, Lebiere C: *The Cascade-Correlation Learning Architecture.* Carnegie Mellon University; 1991. Masters from School of Computer Science.
35. Templeton A. Epistasis and complex traits. Wade, M., Broadie, B III, and Wolf, J. 41-57. 2000. Oxford, Oxford University Press. Epistasis and the Evolutionary Process.
36. Li W, Reich J. A complete enumeration and classification of two-locus disease models. Hum.Hered. 50, 334-349. 2000.
37. Moore J, Hahn L, Ritchie M, Thornton T, White B. Application of genetic algorithms to the discovery of complex models for simulation studies in human genetics. Langdon, WB, Cantu-Paz, E, Mathias, K, Roy, R, Davis, D, Poli, R, Balakrishnan, K, Honavar, V, Rudolph, G, Wegener, J, Bull, L, Potter, MA, Schultz, AC, Miller, JF, Burke, E, and Jonoska, N. Proceedings of the Genetic and Evolutionary Algorithm Conference. 1150-1155. 2002. San Francisco, Morgan Kaufman Publishers.
38. Anderson J: *An Introduction to Neural Networks*. Cambridge, Massachusetts: MIT Press; 1995.
39. Culverhouse R, Suarez BK, Lin J, Reich T: A perspective on epistasis: limits of models displaying no main effect. *Am J Hum Genet* 2002, 70: 461-471.
40. Frankel W, Schork N. Who's afraid of epistasis? Nat.Genet. 14, 371-373. 1996.

Obtaining Biclusters in Microarrays with Population-Based Heuristics

Pablo Palacios[1], David Pelta[2], and Armando Blanco[2]

[1] Dept. de Ingeniería Electrónica, Sistemas Informáticos y Automática, Universidad de Huelva, Campus UIB - Huelva
pablo.palacios@diesia.uhu.es

[2] Depto. de Ciencias de la Computación e I.A., Calle Periodista Daniel Saucedo Aranda s/n, Universidad de Granada, 18071 Granada, Spain
{dpelta, armando}@decsai.ugr.es

Abstract. In this article, we shall analyze the behavior of population-based heuristics for obtaining biclusters from DNA microarray data. More specifically, we shall propose an evolutionary algorithm, an estimation of distribution algorithm, and several memetic algorithms that differ in the local search used.

In order to analyze the effectiveness of the proposed algorithms, the freely available yeast microarray dataset has been used. The results obtained have been compared with the algorithm proposed by Cheng and Church.

Both in terms of the computation time and the quality of the solutions, the comparison reveals that a standard evolutionary algorithm and the estimation of distribution algorithm offer an efficient alternative for obtaining biclusters.

1 Introduction

One of the research fields which has aroused the greatest interest towards the end of the 20th century and whose future is expected to be as equally promising in the 21st century is the study of an organism's genome or genomics.

By way of a brief history, it was Gregor Mendel who defined the gene concept in his research as the element where information about hereditary characteristics is to be found. At a later stage, Avery, McCleod and McCarty demonstrated that an organism's genetic information stems from a macromolecule called deoxyribonucleic acid (DNA); it was later discovered that genetic information located in specific areas of the DNA (the genes) enabled protein synthesis; this was followed by the sequencing of the genome of certain organisms (including humans). This and future consequences awakened a great deal of interest among scientists.

Since proteins are responsible for carrying out cellular functions, cellular functioning therefore depends on the proteins synthesized by the genes, and is determined by regulation of protein synthesis (gene expression) and control of its activity.

The process whereby the approximately 30,000 genes in the human genome are expressed as proteins involves two steps: 1) the DNA sequence is transcribed in messenger RNA sequences (mRNA); and 2) the mRNA sequences are in turn translated into amino acid sequences which comprise the proteins.

Measuring the mRNA levels provides a detailed vision of the subset of genes which are expressed in different types of cells under different conditions. Measuring these levels of gene expression under different conditions helps explore the following aspects (among others) in greater depth: a) The function of the genes, b) How several genes interact and c) How different experimental treatments affect cell function.

Recent advances in array-based methods enable expression levels of thousands of genes to be measured simultaneously. These measurements are obtained by quantizing the mRNA hybridization with a cDNA array, or oligonucleotide probes fixed in a solid substance.

Technological advances in the development of cDNA arrays simultaneously produce an amazingly large quantity of data relating to the transcription levels of thousands of genes and in specific conditions. For knowledge extraction (function of the genes, implication of certain genes in specific illnesses, etc.), researchers use consolidated methodologies and specific ones are being developed. However, although the results obtained so far are getting better, there is still room for improvement.

2 Gene Expression Matrices

In a gene expression matrix, the rows represent genes and the columns represent samples, and each cell contains a number which characterizes the expression level of a particular gene in a particular sample.

Like most experimental techniques, microarrays measure the final objective indirectly through another physical quantity, for example the relative abundance of mRNA through the fluorescence intensity of the *spots* in an array.

Microarray-based techniques are still a long way from providing the exact quantity of mRNA in a cell. The measurements are naturally relative: essentially we can compare the expression levels of one gene in different samples or different genes in one sample, so that it is necessary to apply a suitable normalization to enable comparisons between data. Moreover, as the value of the microarray-based gene expression can be considerably greater according to the reliability and limitations of a particular microarray technique for certain types of measurements, data normalization is a key issue to consider.

Once we have constructed the gene expression matrix, the second step is to analyze it and attempt to obtain information from it.

In this work we shall use the *biclustering* concept introduced by Hartigan [6] to capture the degree of similarity between a *subset* of elements within a *subset* of attributes. Church applied this technique on DNA *microarrays* [3].

The advantage of biclustering as opposed to traditional clustering when applied to the field of microarrays lies in its ability to identify *groups* of genes that show similar activity patterns under a *specific subset* of the experimental conditions. Therefore, biclustering approaches are the key technique to use when one or more of the following situations applies [7]:

1. Only a small set of the genes participates in a cellular process of interest.
2. An interesting cellular process is active only in a subset of the conditions.
3. A single gene may participate in multiple pathways that may or not be coactive under all conditions.

Besides, the biclusters should not be exclusive and/or exhaustive: A gene / condition should be able to belong to more than one cluster or to no cluster at all and be grouped using a subset of conditions/genes.

2.1 Biclustering Techniques

In this section, we briefly present some representative strategies in literature for obtaining biclusters.

Cheng and Church in [3], proposed a set of heuristic algorithms whose function, beginning with the complete matrix, is based on the execution of iterative stages: deletion and addition of rows in order to obtain biclusters.

The *FLOC* algorithm (Flexible Overlapped Clustering) [16], is a variant of previous work, which performs an iterative process from an initial set of biclusters in an attempt to improve their overall quality. In each iteration, a row or column is added or deleted from each bicluster in order to produce the ideal set of biclusters in terms of having the greatest similarity. The algorithm finishes when there is no improvement in the overall quality of the previous set of biclusters.

The *CLICK* algorithm is based on the construction of bipartite graphs [13]. This algorithm uses the following three stages to obtain biclusters: a) Identify regions which may contain biclusters; b) Identify the biclusters and c) Refine biclusters to their minimum size.

The *double conjugated clustering* algorithm [2], where the search for biclusters is performed on two different search spaces: one comprising the genes, and the other the experiments. In this way, a clustering algorithm is applied to each space independently. In order to join the clusters induced in both processes, two functions are defined which enable one node from one space to be converted into the conjugated node of the other space, and vice versa. The final adjustment between both search spaces is obtained by means of a new clustering process to correct the clusters conjugated in the other space.

The *pCluster* algorithm [14] uses a more general similarity type. Two objects therefore belong to the same cluster if they display a similar pattern for a subset of dimensions. This enables biclusters to be discovered with elements which comply with the same pattern although they are not close to each other. Discovering these biclusters is essential when revealing gene behavior.

Finally, the *pMafia* algorithm [10] consists of a parallel implementation of a grid algorithm [12, 15] adapted for biclusters. Each dimension is divided into small intervals of a fixed size called "windows". During the clustering process, when two adjacent windows are similar, they merge and a new one is created. The algorithm uses the parallelism to reduce the computation time.

The last years have shown an increasing interest in this field. We suggest the interested reader to check out the excellent survey by Madeira and Oliveira [7].

2.2 Measuring the Quality of a Bicluster

Below, we shall define the main concepts which form the basis of the residue-based bicluster induction methods towards which we have directed our research.

Definition: Let D be a gene expression matrix, of the size $n \times m$ ($D_{n \times m}$), where the set of rows $F = \{G_1, G_2, ..., G_n\}$ represents the genes and the set of columns $R = \{E_1, E_2, ..., E_m\}$ represents the conditions or experiments. Each element d_{ij} in the matrix matches the expression level (absolute or relative) of gene G_i in experiment E_j.

Definition: Given a gene expression matrix $D_{n \times m}$, a bicluster is a pair (I, J), where $I \subseteq \{1, ..., n\}$ is a subset of rows of F and $J \subseteq \{1, ..., m\}$ is a subset of columns of R, in which the genes G_i with $i \in I$ behave in a similar way.

Definition: Given a bicluster (I, J), the residue (r_{ij}) of an element d_{ij} of the bicluster is calculated according to Equation 1.

$$r_{ij} = d_{ij} - d_{iJ} - d_{Ij} + d_{IJ} \tag{1}$$

where

$$d_{iJ} = \frac{\sum_{j \in J_i} d_{ij}}{|J_i|} \tag{2}$$

$$d_{Ij} = \frac{\sum_{i \in I_j} d_{ij}}{|I_j|} \tag{3}$$

$$d_{IJ} = \frac{\sum_{i \in I, j \in J} d_{ij}}{|I| \cdot |J|} \tag{4}$$

The residue is an indicator of the degree of coherence of an element in relation to the remaining elements in the bicluster, additionally providing the bias of the objects and the relevant attributes. Therefore, the smaller the value of the residue, the greater the coherence.

Definition: Given a bicluster (I, J), the residue (r_{IJ}) of the bicluster can be obtained from Equation 5, where r_{ij} is the residue of the element d_{ij} and v_{IJ} is the volume of the bicluster.

$$r_{IJ} = \frac{\sum_{i \in I, j \in J} |r_{ij}|}{v_{IJ}} \tag{5}$$

In order to determine the overall quality of a bicluster, its residue is defined as the mean of the residues of all its elements. This mean could be arithmetic, geometric, etc. Here, we applied the arithmetic mean.

3 Proposed Population Based Techniques

In this section, we shall describe the particular characteristics of the genetic algorithm (GA), the memetic algorithms (MA), and the estimation of distribution algorithm (EDA) which have been implemented.

3.1 Genetic Algorithm

As the simplest population based technique, we shall implement a classical genetic algorithm, with elitism. The main characteristics are described next.

Codification: The solution's representation is as follows: given a data matrix D of size $n \times m$, the biclusters are encoded in a vector of size $n + m$, where the first n positions represent the rows and the last m positions represent the columns of the bicluster. Each position of the vector can have one of two values (1 or 0) indicating whether the corresponding row or column is to be found (1) or not (0) in the bicluster.

Selection: Baker's stochastic universal sampling was chosen as the selection method. This is a roulette wheel method with slots which are sized according to the fitness of each chromosome.

Crossover: the uniform operator is used: given two parents, the offspring keep the values common to both of them, while every other value is randomly taken from any of the parents.

Mutation: a *BitFlip* mechanism was chosen: given an individual in the population, one of its bit values is changed for its complementary one.

Fitness: is measured as the residue associated to the bicluster represented by a given individual.

Restart: For the restart strategy, we have chosen to move the best individual to the new population. In addition, 20 % of the new population will be the best individual in the current mutated generation and the rest shall be generated randomly. This restart shall be applied when 10 % of the generations to be made have taken place with no change in the best element in the population.

3.2 Memetic Algorithms

Memetic algorithms have been studied from practical and theoretical points of view since 15 years ago [5] and, while very complex strategies are being developed, in their simplest form they can still be considered as a genetic algorithm hybridized with a local search operator.

Here, and departing from the genetic algorithm described before we implemented several basic memetic algorithms using two different local search techniques, namely:

- **K-opt:** the chromosomes can also be seen as a permutation of the rows and columns of the bicluster so that we could apply *k-opt* movements on

them, which in particular would consist in exchanging with each other k bits of a given solution. If this exchange leads to an improvement, a new k-opt movement would be undertaken and this process would be continued until no improvement is made in certain number of trials.
We constructed 4 different memetic schemes for values of $k \in \{2, 3, 4, 5\}$.

– **Taboo Search (TS):** a very basic TS strategy is used. The neighborhood is sampled using the mutation operator described for the GA.

3.3 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDA) were described for the first time by H. Muehlenbein and G. Paab [8]. In EDAs, the solution space is represented by means of a probability distribution associated with the individuals selected in each generation and not with a population. This probability distribution is calculated from a set of individuals selected from the previous generation. Once obtained, it is sampled in order to generate descendants so that neither the mutation nor the crossover is applied in the EDAs.

The easiest way to calculate the probability distribution consists in considering all the variables of interest to be independent. So, the probability estimation is converted into the product of the marginal probabilities of n variables as:

$$p(x) = \Pi_{i=1}^{n} p(x_i) \tag{6}$$

Different approximations to the methodology can be found, including: the univariate marginal distribution algorithm [9], population-based incremental learning [1] and the compact genetic algorithm [4].

In this work we shall focus on the Univariate Marginal Distribution Algorithm (UMDA, in what follows). UMDA maintains a population of N individuals, to which a selection method is applied in order to create a new population. From this new population, the frequencies of each gene are obtained and used to generate a new population of N individuals. This mechanism for generating the population is a type of crossover operator which replaces the traditional GA crossover operator.

The process in detail is as follows:

1. Choose the M best individuals in the current population which are in the set $D_{l-1}^{S_e}$.
2. Estimate the probability distribution of the current population using Eq. 7.
3. Generate a new population of N individuals from the probability distribution which is stored in the set $D_l^{S_E}$.

The probability distribution is expressed as the product of invariable marginal probabilities, which is estimated from the marginal frequencies:

$$p_l(x_i) = \frac{\sum_{j=1}^{M} \delta_j(X_i = x_i | D_{l-1}^{S_e})}{M} \tag{7}$$

where $\delta_j(X_i = x_i|D^{S_e}_{l-1})$ is 1 for the j^{th} individual in M if the value of gene X_i is equal to x_i, in any other case it will be 0.

The outline of our UMDA-based algorithm can be seen in Algorithm 1.

Algorithm 1. Proposed Estimation of Distribution Algorithm

```
1. Generate initial population of size $N$
2. Do $itersNow = 0$
3. While $itersNow < maxIters$
    (a) Choose $\frac{N}{2}$ individuals
    (b) Estimate the probability distribution of the $M$ individuals
    (c) Sample the distribution in order to obtain new individuals
    (d) Replace the old population with the new one
4. End While
5. Return best individual
6. End
```

4 Experiments

In order to evaluate and analyze the implemented algorithms, the yeast expression data set has been used, comprising 17 experiments (columns) on 2900 genes (rows). This gene expression data set was chosen since it is one of the most used in literature by the majority of experts in this field, thereby enabling our results to be compared.

The results obtained with the proposed tools have been compared using the algorithm proposed by Church in [3] as a reference algorithm.

Following an empirical study, the following parameters were fixed: a population of 200 individuals and 200 generations. The crossover and mutation probabilities were fixed at 0.8 and 0.6, respectively.

Each algorithm was executed 30 times, and the seed of the random number generator was changed in each execution. At the end of each execution, the best bicluster found was recorded.

4.1 Results

This section includes the results obtained by the proposed algorithms and the reference algorithm. We also performed a random sampling of biclusters in order to check the expected residue value for a random bicluster.

Table 1 shows the corresponding residues for the best and worst biclusters, the mean and typical deviation on 30 executions, and also an indication of the time taken for each execution. The average size of the resulting biclusters are also displayed. Results for Reference algorithm were taken over 100 bicluster while 10^4 random solutions were generated.

Figure 1 shows the histograms of residue (a), rows (b) and columns (c) of the best biclusters found by every algorithm. Results for k-opt for $k \geq 3$ were omitted for visualization purposes (although they were extremely similar to those of 2-opt). These figures give us a global view of the whole set of best biclusters available, enabling to quickly check out the region of the search space covered by every strategy.

Several aspects can be highlighted from the table and the histograms. First one is that the GA achieves very good residue values despite the simplicity of its components' definitions. On average, the biclusters generated are quite big (825 rows and 8 columns).

The joint use of GA with local search, leading to different memetic schemes, does not seem to be useful. The simpler local search schemes (k-opt) increase the residue while the average number of rows in the bicluster is significantly reduced. In this case, and given the standard deviation values, it is clear that there is a problem of convergence which is independent of the k value. Moreover, no statistical differences were detected for different values of k.

As the complexity of the local search is increased, from 2-opt to TS, the residue values also increase. This becomes clear if we look at the corresponding histogram. In turn, the sizes of the biclusters obtained are slightly higher than those obtained by k-opt.

The EDA strategy achieves the lowest average residue value, while the corresponding bicluster sizes are about 200 rows and 8 columns. The average residue for the reference algorithm is almost three times higher than that of EDA, while the biclusters are smaller on average(although the number of columns is increased from 8 to 12). The reference algorithm presents the highest variability in residue, number of rows and columns (this is clearly seen in the histograms).

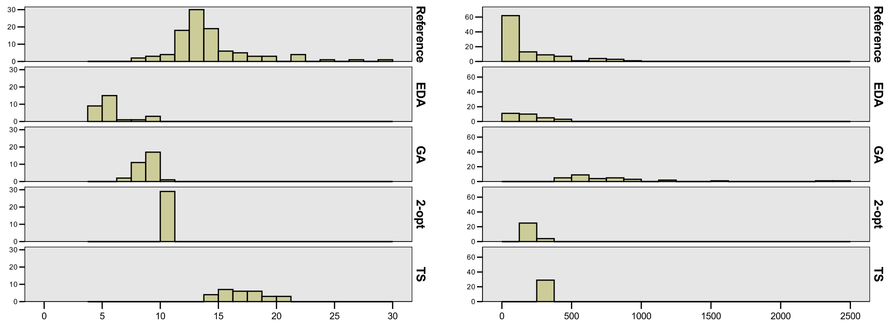
In order to determine what differences in terms of residue are significant, a Kruskal-Wallis test was performed. The test reveals significant differences among the median of the residues of the algorithms. Then, pairwise U Man-Witney non parametrical test were performed and they confirm that the differences among the algorithms were significant.

Another element to analyze is the computational time used. The faster algorithm, and the best one on bicluster quality, is EDA, followed by the GA. The addition of local search to GA increases the computational time considerably while not having the same counterpart in biclusters quality. The Church's algorithm has quite acceptable running times.

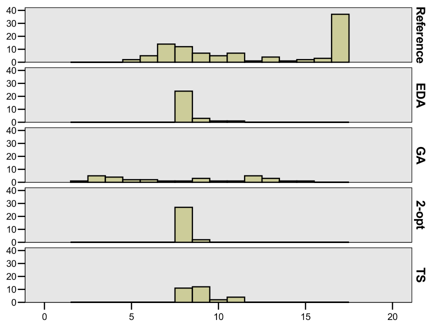
In Fig. 2 we plot the volume of the best solutions (calculated as $rows \times columns$) against residue for algorithms GA, EDA and Reference). This plot reveals several things. First one is the existence of many alternative solutions with similar residue. See for example the vertical range for residue between 5-10. This fact is most notably for GA and EDA. In second place we can see that the Reference algorithm is able to obtain very similar solutions in size while very different in residue. Both facts clearly encourage the use of population based techniques that allow to simply manage a set of solutions of diverse characteristics.

Table 1. Statistical Values of residue and size of the biclusters found by every algorithm. Time is in minutes per run.

	Residue			Average Size		
Algorithm	Avg (sd)	Best	Worst	Rows (sd)	Cols (sd)	Time
GA	8.83 (0.81)	7.30	11.82	825.19 (488.32)	7.94 (4.11)	5
EDA	5.72 (1.45)	4.81	9.87	213.28 (109.74)	8.28 (0.70)	3
GA+2-opt	10.46 (0.04)	10.35	10.50	235.62 (9.95)	8.07 (0.26)	10
GA+3-opt	10.45 (0.05)	10.36	10.50	241.59 (10.14)	8.07 (0.26)	14
GA+4-opt	10.45 (0.05)	10.37	10.49	243.48 (11.71)	8.14 (0.35)	15
GA+5-opt	10.47 (0.04)	10.33	10.50	240.83 (15.67)	8.04 (0.21)	17
GA+TS	17.07 (1.94)	13.90	20.68	280.20 (10.99)	8.00 (0.12)	14
Church	14.19 (3.59)	7.77	29.78	166.70 (226.37)	12.09 (4.40)	5-10
Random	23.51 (2.60)	5.75	34.26	1407.30 (841.16)	9.48 (4.60)	–



(a) (b)



(c)

Fig. 1. Histograms of residue (a) , row (b) and column's (c) values of the best biclusters obtained by every algorithm. Results for k-opt for $k \geq 3$ were omitted. TS and 2-opt stands for the memetic algorithm using such local search scheme.

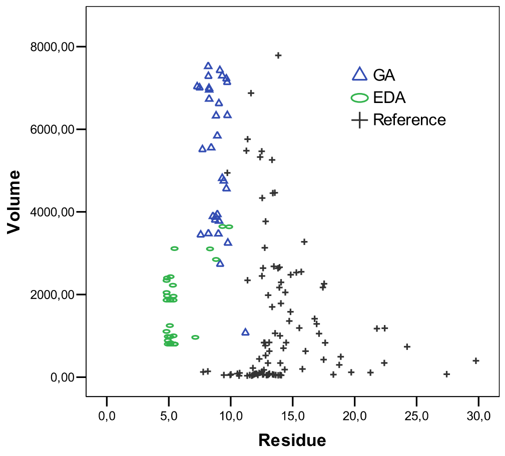


Fig. 2. Volume (*row* × *column*) vs Residue of the best biclusters found by EDA, GA and Reference Algorithm

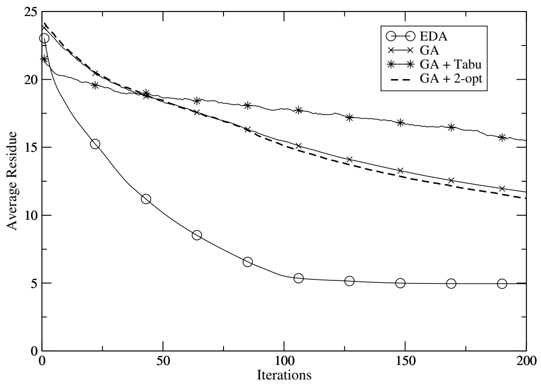


Fig. 3. Time vs Residue EDA, GA, GA+TS and GA + 2-opt Algorithms

Finally, Fig. 3 shows the evolution of the average residue over the time for typical runs of EDA, GA, GA+TS and GA + 2-opt. The faster convergence is achieved by EDA; given that no restart mechanism is included, EDA becomes stagnated during the last half of the time available. The curves for GA and GA+2-opt are pretty similar: both algorithms show a continuous but very slow convergence. Also, GA+TS is the worst method: it seems like the algorithm can not made the population to converge. We have two hypothesis for these behaviors: first one there may be a problem in the local search parameters; second one may be related with the fact that a small change in the genotype can give raise to a big change in the phenotype and, when this fact occurs, the use of local search is not recommendable. Both situations are under study but we suspect the second reason may be more relevant.

5 Conclusions and Future Research

The population based techniques tested here showed as robust and efficient strategies for coping with the obtention of biclustering in microarray matrices.

More specifically, the simple EDA implemented was able to obtain better solutions than the other algorithms (including the reference one) while using less computational time.

We are aware that the analysis of biclustering results is somehow controversial because it is not clear what makes a bicluster "good" or not. In principle, it seems desirable for the biclusters to be as large as possible and to maintain low residue levels. This would indicate that a high number of genes have been identified with a similar expression profile in a large number of conditions.

Also, Aguilar [11] proved that the mean squared residue is not precise enough, from the mathematical point of view, to discover shifting and scaling patterns simultaneously and he pointed out that the characterization of an objective function H that allows to detect both patterns simultaneously would be very beneficial. To the best of our knowledge, such function is still not available.

A byproduct of using these population-based techniques is that, at the end of each run, we have available a set of high quality solutions. This is extremely important because the ultimate goal is to obtain a set of genes "biologically" related, so the optimization point is somehow secondary. In this context, the use of more complex strategies like memetic algorithms without having problem specific operators seems not to be recommendable.

Acknowledgments

This work was supported in part by projects TIC2003-09331-C02-01 and TIN2005-08404-C04-01 from the Spanish Ministry of Science and Technology.

References

1. S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russel, editors, *The Int. Conf. on Machine Learning*, pages 38–46. Morgan Kaufmann Publishers, 1995. San Mateo, CA.
2. S. Busygin, G. Jacobsen, and E. Kramer. Double conjugated clustering applied to leukemia microarray data. *SIAM ICDM, Workshop on clustering high dimensional*, 2002.
3. Y. Cheng and G. Church. Biclustering of expression data. *8th International Conference on Intelligent System for Molecular Biology*, pages 93–103, 2001.
4. G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *IEEE-EC*, 3(4):287, November 1999.
5. W. Hart, N. Krasnogor, and J. Smith, editors. *Recent Advances in Memetic Algorithms*. Studies in Fuzziness and Soft Computing. Physica-Verlag, 2004.
6. J. Hartigan. *Clustering Algorithms*. John Wiley, 1975.
7. S. Madeira and A. Olivera. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on computational biology an bioinformatics.*, 1(1):24–45, 2004.

8. H. Muehlenbein and G. Paab. From recombination of genes to the estimation of distributions. In *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature*, volume IV, pages 178–187. PPSN, 1996.
9. H. Muhlenbein. Evolutionary computation: The equation for response to selection and its use for prediction. *Evolutionary Computation*, (5):303–346, 1998.
10. H. Nagesh, S. Goil, and A. Choudhary. pmafia: A scalable parallel subspace clustering algorithm for massive data sets. *International Conference on Parallel Processing*, pages 477–, 2000.
11. J. A. Ruiz. Shifting and scaling patterns from gene expression data. *Bioinformatics*, 21(20):3840–3845, 2005.
12. E. Schikuta. Grid-clustering: An efficient hierarchical clustering method for very large data sets. *In Proc.13th Int. Conf. Pattern Recognition, IEEE Computer Society.*, 2:101–105, 1996.
13. R. Sharan and R. Shamir. Click: A clustering algorithm with applications to gene expression analysis. *Proceedings of the Eighth International Conference on Intelligent Systems*, pages 307–316, 2000.
14. H. Wang, W. Wang, J. Yang, and P. Yu. Clustering by pattern similarity in large data sets. *SIGMOD Conference*, 2002.
15. W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. *In Proc. 23rd Conf. Very Large Databases*, pages 186–195, 1997.
16. J. Yang, H. Wang, W. Wang, and P. Yu. Improving performance of bicluster discovery in a large data set. *Proceedings of the 6th ACM International Conference on Research in Computational Molecular Biology (RECOMB)*, 2002.

Multiple Sequence Alignment Based on Set Covers

Alexandre H.L. Porto and Valmir C. Barbosa[⋆]

Universidade Federal do Rio de Janeiro,
Programa de Engenharia de Sistemas e Computação, COPPE,
Caixa Postal 68511, Rio de Janeiro - RJ 21941-972, Brazil
valmir@cos.ufrj.br

Abstract. We introduce a new heuristic for the multiple alignment of a set of sequences. The heuristic is based on a set cover of the residue alphabet of the sequences, and also on the determination of a significant set of blocks comprising subsequences of the sequences to be aligned. These blocks are obtained with the aid of a new data structure, called a suffix-set tree, which is constructed from the input sequences with the guidance of the residue-alphabet set cover and generalizes the well-known suffix tree of the sequence set. We provide performance results on selected BAliBASE amino-acid sequences and compare them with those yielded by some prominent approaches.

Keywords: Multiple sequence alignment; Set covers.

1 Introduction

The multiple alignment of biological sequences is one of the most important problems in computational molecular biology. It has applications in many different important domains, such as the analysis of protein structure and function, the detection of conserved patterns and domain organization of a protein family, evolutionary studies based on phylogenetic analysis, and database searching for new members of a protein family.

The problem of multiple sequence alignment can be stated in the following manner. Let $s_1, \ldots, s_k$, with $k \geq 2$, be sequences of lengths $n_1, \ldots, n_k$, respectively, over a residue alphabet R. An alignment $\mathcal{A}$ of these sequences is a $k \times l$ matrix such that $\mathcal{A}[i, j]$, for $1 \leq i \leq k$ and $1 \leq j \leq l$, is either a character in R or the special character that we call a gap character. In $\mathcal{A}$, fixing i and varying j from 1 through l must reproduce s_i exactly if gap characters are skipped. Fixing j, in turn, yields k characters that are said to be aligned in $\mathcal{A}$, of which at least one must be in R. Note, then, that $\max\{n_1, \ldots, n_k\} \leq l \leq n_1 + \cdots + n_k$.

The goal of the multiple sequence alignment problem is to determine the most biologically significant alignment of $s_1, \ldots, s_k$. Finding this alignment requires an objective function to associate a score with each possible alignment, and in this

⋆ Corresponding author.

case the multiple sequence alignment problem is to find an alignment, known as the optimal alignment, that maximizes the objective function. There exist many different objective functions that can be used, but none of them guarantees that the corresponding optimal alignment is the most biologically significant alignment of the sequences. It follows from the definition of an alignment that the number of different alignments of a given set of sequences is exponentially large; in fact, the multiple sequence alignment problem is known to be at least NP-hard [1]. Feasible approaches to solve the problem are then all of a heuristic nature.

In this paper we describe a new heuristic that is based on set covers of the residue alphabet R. Such a set cover is a collection of subsets of R whose union yields R precisely. The idea behind the use of a set cover is that each subset can be made to contain all the residues from R that possess certain structural or physicochemical properties in common. The most familiar scenario for the use of set covers is the case of R as a set of amino acids, so henceforth we concentrate mainly on multiple protein alignments. Set covers of an amino-acid alphabet have been studied extensively (e.g., [2]).

Set covers lie at the heart of the new heuristic. In essence, what they do is to allow the introduction of a new data structure, called a suffix-set tree, that generalizes the well-known suffix tree of a set of sequences and can be used in the determination of subsequence blocks that ultimately give rise to the desired alignment. In general terms, this is the same approach as some others in the literature, but our use of set covers as its basis provides a fundamentally more direct link between relevant properties shared by groups of residues and the resulting alignment.

The following is how the remainder of the paper is organized. In Section 2 we introduce our new data structure and in Section 3 describe our new approach to multiple sequence alignment. Then we proceed in Section 4 to the presentation of computational results of the new method as compared to some of its most prominent competitors, and finalize with conclusions in Section 5.

2 Suffix-Set Trees

In this section we describe our new data structure. It is a generalization of the well-known suffix tree of a set of sequences, which is one of the most important data structures in the field of pattern recognition. Such a suffix tree has $O(n_1 + \cdots + n_k)$ nodes and can be constructed in $O(n_1 + \cdots + n_k)$ time [3].

Suffix trees can be applied to many problems, but their principal application in computational molecular biology is to assist the algorithms that try to obtain blocks comprising biologically significant subsequences of a set of sequences, known as the motifs of that set. These motifs, in the case of proteins, encode structural or functional similarities; in the case of nucleic acids, they encode mainly the promoter regions.

Let $\mathcal{C} = \{C_1, \ldots, C_p\}$ be a set cover of R, and let $\Sigma_{\mathcal{C}} = \{\alpha_1, \ldots, \alpha_p, \alpha_\$\}$ be a new alphabet having a character α_i for each $C_i \in \mathcal{C}$ and a further character

$\alpha_\$$ possessing a function similar to the terminator used in suffix trees. Like the suffix tree, our new data structure is also a rooted tree; it has edges labeled by sequences of characters from $\Sigma_\mathcal{C}$ and nodes labeled by indices into some of $s_1, \ldots, s_k$ to mark suffix beginnings. We call it a suffix-set tree and it has the following properties:

– The first character of the label on an edge connecting a node to one of its children is a different character of $\Sigma_\mathcal{C}$ for each child.
– Each nonempty suffix of every one of the k sequences is associated with at least one leaf of the tree; conversely, each leaf of the tree is associated with at least one nonempty suffix of some sequence (if more that one, then all suffixes associated with the leaf have the same length). Thus, each leaf is labeled by a set like $\{(i_1, j_1), \ldots, (i_q, j_q)\}$ for some $q \geq 1$, where (i_a, j_a), for $1 \leq a \leq q$, $1 \leq i_a \leq k$, and $1 \leq j_a \leq n_{i_a}$, indicates that the suffix $s_{i_a}[j_a \mathinner{..} n_{i_a}]$ of s_{i_a} is associated with the leaf.
– Let v be a node of the tree. The label of v is the set $\{(i_1, j_1), \ldots, (i_q, j_q)\}$ that represents the q suffixes collectively associated with the leaves of the subtree rooted at v. If $\alpha_{c_1} \cdots \alpha_{c_r}$ is the concatenation of the labels on the path from the root of the tree to v, excluding if necessary the terminal character $\alpha_\$$, then $\alpha_{c_1} \cdots \alpha_{c_r}$ is a common representation of all prefixes of length r of the suffixes associated with the leaves of the subtree rooted at v. If $s_{i_a}[j_a \mathinner{..} n_{i_a}]$ is one of these suffixes, then for $1 \leq b \leq r$ we have $s_{i_a}[j_a + b - 1] \in C_{c_b}$ (that is, the bth character of the suffix is a member of C_{c_b}).

Note that, when $\mathcal{C}$ is a partition of R into singletons, the suffix-set tree becomes the familiar suffix tree of $s_1, \ldots, s_k$. In order to see this, it suffices to identify for each character in each sequence the member C_i of $\mathcal{C}$ to which it belongs, and then substitute α_i for that character. We show in Figure 1 a suffix-set tree for $R = \{A, C, G, T\}$, $C_1 = \{A, G, T\}$, $C_2 = \{C, T\}$, $s_1 = AGCTAG$, and $s_2 = GGGATCGA$.

In the strategy to be described in Section 3 we do not construct the suffix-set tree to completion, but rather only the portion of the tree that is needed to represent all suffix prefixes of length at most M (a fixed parameter). Clearly, the number of nodes in the tree is $O(p^M)$, therefore polynomial in p given the fixed parameter M. It is relatively simple to see that the tree can be constructed in $O\left(p^{M+1}(n_1 + \cdots + n_k) + p^{M+2}|R|^2\right)$ time.

3 Alignments from Set Covers

For $2 \leq k' \leq k$, let $t_1, \ldots, t_{k'}$ be subsequences of $s_1, \ldots, s_k$ such that each t_a is a subsequence of a different s_{i_a}, with $1 \leq a \leq k'$ and $1 \leq i_a \leq k$. We call $\{t_1, \ldots, t_{k'}\}$ a block. If $\mathcal{A}'$ is an alignment of $t_1, \ldots, t_{k'}$ having l' columns, then the score of $\mathcal{A}'$, denoted by $S(\mathcal{A}')$, is a function, to be introduced shortly, of

$$T(\mathcal{A}') = \sum_{c=1}^{l'} \sum_{a=1}^{k'-1} \sum_{b=a+1}^{k'} Q\left(p_a^c, p_b^c, \mathcal{A}'[a, c], \mathcal{A}'[b, c]\right), \tag{1}$$

where p_a^c is the position in s_{i_a} of the rightmost character of t_a whose column in $\mathcal{A}'$ is no greater than c ($p_a^c = 0$, if none exists), and similarly for p_b^c. In (1), $Q\left(p_a^c, p_b^c, \mathcal{A}'[a,c], \mathcal{A}'[b,c]\right)$ is the contribution to $T(\mathcal{A}')$ of aligning the two characters $\mathcal{A}'[a,c]$ and $\mathcal{A}'[b,c]$ given p_a^c and p_b^c. If both $\mathcal{A}'[a,c]$ and $\mathcal{A}'[b,c]$ are gap characters, then we let $Q\left(p_a^c, p_b^c, \mathcal{A}'[a,c], \mathcal{A}'[b,c]\right) = 0$. Otherwise, the value of $Q\left(p_a^c, p_b^c, \mathcal{A}'[a,c], \mathcal{A}'[b,c]\right)$ is determined through a sequence of two steps.

The first step is the determination of the combined number of optimal global and local pairwise alignments of s_{i_a} and s_{i_b} that go through the p_a^c,p_b^c cell of the dynamic-programming matrix. In what follows, we split this number into three others, and let each of $U_1(p_a^c, p_b^c), \ldots, U_3(p_a^c, p_b^c)$ be either a linearly normalized version of the corresponding number within the interval $[1, L]$ for L a parameter, if that number is strictly positive, or 0, otherwise. We use $U_1(p_a^c, p_b^c)$ in reference to the case of optimal alignments through the p_a^c,p_b^c cell that align $s_{i_a}[p_a^c]$ to $s_{i_b}[p_b^c]$, $U_2(p_a^c, p_b^c)$ for alignments of $s_{i_a}[p_a^c]$ to a gap character, and $U_3(p_a^c, p_b^c)$ for alignments of $s_{i_b}[p_b^c]$ to a gap character.

The second step is the determination of $Q\left(p_a^c, p_b^c, \mathcal{A}'[a,c], \mathcal{A}'[b,c]\right)$ itself from the quantities $U_1(p_a^c, p_b^c), \ldots, U_3(p_a^c, p_b^c)$. If $U_1(p_a^c, p_b^c) = \cdots = U_3(p_a^c, p_b^c) = 0$ (no optimal alignments through the p_a^c,p_b^c cell), then $Q\left(p_a^c, p_b^c, \mathcal{A}'[a,c], \mathcal{A}'[b,c]\right) = -L$. Otherwise, we have the following cases to consider, where $z \in \{1,2,3\}$ selects among U_1, U_2, and U_3 depending, as explained above, on $\mathcal{A}'[a,c]$ and $\mathcal{A}'[b,c]$:

– If $U_z(p_a^c, p_b^c) > 0$, then $Q\left(p_a^c, p_b^c, \mathcal{A}'[a,c], \mathcal{A}'[b,c]\right) = U_z(p_a^c, p_b^c)$.
– If $U_z(p_a^c, p_b^c) = 0$, then

$$Q\left(p_a^c, p_b^c, \mathcal{A}'[a,c], \mathcal{A}'[b,c]\right) = -\min{}^{+}\left\{U_1(p_a^c, p_b^c), U_2(p_a^c, p_b^c), U_3(p_a^c, p_b^c)\right\},$$

where we use $\min^+$ to denote the minimum of the strictly positive arguments only.

What this second step is doing is to favor the alignment of $\mathcal{A}'[a,c]$ to $\mathcal{A}'[b,c]$ in proportion to its popularity in optimal pairwise alignments of s_{i_a} and s_{i_b}, and similarly to penalize it—heavily when cell p_a^c, p_b^c is part of no optimal pairwise alignment, less so if it is but not aligning $\mathcal{A}'[a,c]$ to $\mathcal{A}'[b,c]$.

Finally, the function that yields $S(\mathcal{A}')$ from $T(\mathcal{A}')$ is designed to differentiate two alignments of different blocks for which T might yield the same value. We do so by subtracting off $T(\mathcal{A}')$ the fraction of $|T(\mathcal{A}')|$ obtained from the average of two numbers in $[0,1]$. The first number is $1-k'/k$ and seeks to privilege (decrease T by a smaller value) the block with the greater number of subsequences. The second number is a function of the so-called identity score of an alignment, that is, the fraction of the number of aligned residue pairs that corresponds to identical residues. If we denote the identity score of $\mathcal{A}'$ by $I(\mathcal{A}')$, then the second number is $1 - I(\mathcal{A}')$ and aims at privileging alignments whose identity scores are comparatively greater. We then have

$$S(\mathcal{A}') = T(\mathcal{A}') - \left(2 - \frac{k'}{k} - I(\mathcal{A}')\right)\frac{|T(\mathcal{A}')|}{2}. \qquad (2)$$

The remainder of this section is devoted to describing our heuristic to obtain a $k \times l$ alignment $\mathcal{A}$ of the sequences $s_1, \ldots, s_k$, given the set cover $\mathcal{C}$ of the

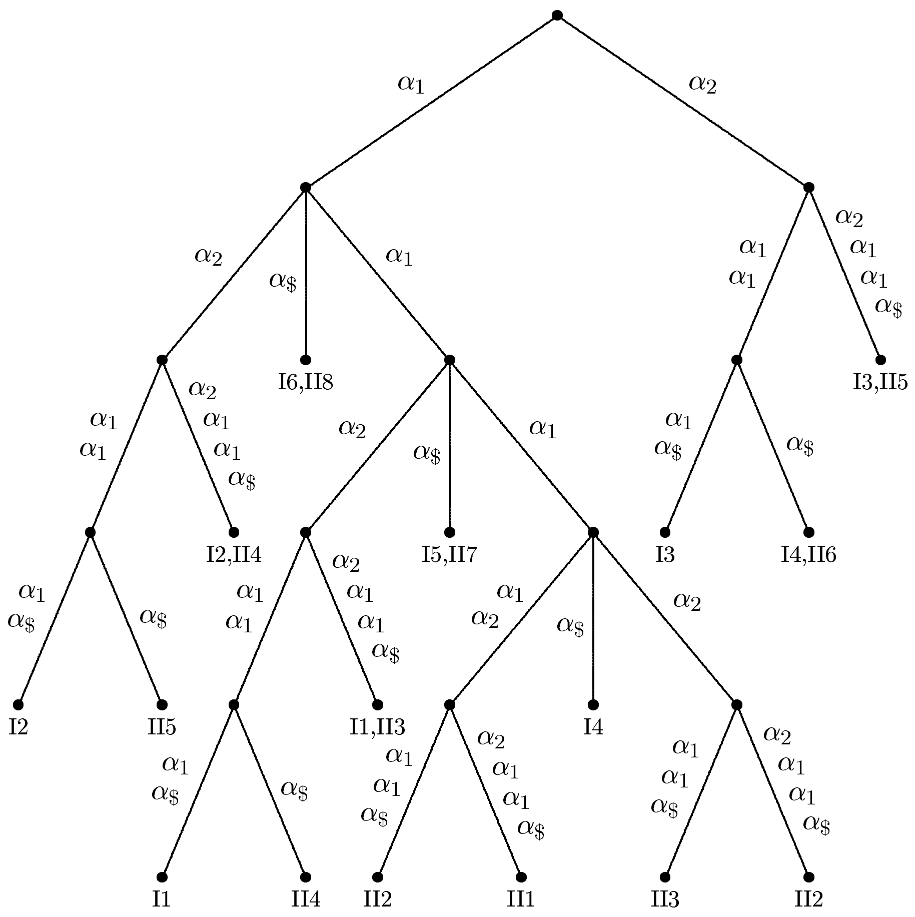


Fig. 1. An example suffix-set tree, node labels shown only for the leaves. Each node label is expressed more compactly than in the text; for example, “I2,II4” stands for the label $\{(1, 2), (2, 4)\}$.

residue alphabet R. The suffix-set tree $\mathcal{T}$ of $s_1, \ldots, s_k$ plays a central role. We first describe how to obtain a suitable set $\mathcal{B}$ of blocks from $s_1, \ldots, s_k$, and then how to obtain $\mathcal{A}$ from $\mathcal{B}$.

3.1 Blocks from Set Covers

We start by creating a set $\mathcal{B}$ of blocks that is initialized to $\emptyset$ and is augmented by the inclusion of new blocks as they are generated. The size of $\mathcal{B}$ is at all times bounded by yet another parameter, denoted by N.

Every node v of $\mathcal{T}$ that is not the root may contribute blocks to $\mathcal{B}$. If $n_v \leq k$ is the number of distinct sequences with suffixes associated with v and $l_v \leq M$ is the common prefix length of all those suffixes, then each block contributed by v is formed exclusively by some of the length-l_v prefixes as its subsequences, totaling at least two and including at most one from each of the n_v sequences.

Let $\mathcal{A}_B$ denote an alignment of block B's subsequences. Block formation for node v proceeds as follows. First the n_v sequences are sorted in nonincreasing order of their numbers of suffixes associated with v and a new block is created for each prefix of the first-ranking sequence. An attempt is then made to add more prefixes to each such block B by visiting the remaining $n_v - 1$ sequences in order and selecting for each one the prefix, if any, that increases $S(\mathcal{A}_B)$ the most when $\mathcal{A}_B$ acquires another row by the addition of that prefix. It is worthy of mention that, as prefixes coalesce into the final form of B, $\mathcal{A}_B$ never contains any gap characters, in which case Q is seen to revert to a simpler form. But notice that the functional form in (2) continues to be effective, not only because of the identity scores, but also because the expansion of B involves comparing alignments that may differ in numbers of rows (a unit difference, to be specific). At the end, all blocks still having one single sequence are discarded.

Once v's contribution to $\mathcal{B}$ is available, it is sorted and then merged into $\mathcal{B}$ (we may think of $\mathcal{B}$ as being internally organized as a sorted list). Both operations seek to retain inside $\mathcal{B}$ those blocks whose alignments' scores are greater. If needed, additional tie-breaking criteria are employed, including those that are already reflected in (2): greater numbers of subsequences and identity scores are preferred.

Now say that two blocks B and B' are such that B is contained in B' if every subsequence of B is itself a subsequence of the corresponding subsequence of B'. Once every non-root node of $\mathcal{T}$ has been considered for contributing to $\mathcal{B}$, the resulting $\mathcal{B}$ is further pruned by the removal of every block that is contained in another of at least the same score. After this, $\mathcal{B}$ undergoes another fix, which is to extend its blocks by the addition of new subsequences so that at the end they all contain exactly one subsequence from each of $s_1, \ldots, s_k$.

The following is how we achieve this extension for block $B \in \mathcal{B}$. We consider the unrepresented sequences one by one in nonincreasing order of their lengths. Then we use a semi-global algorithm to align the current $\mathcal{A}_B$ to a subsequence of the unrepresented sequence under consideration. This algorithm employs the Q function in place of a substitution matrix and gap costs. As B is thus extended, its alignment $\mathcal{A}_B$ changes as well, and may now acquire gap characters for the first time.

The final step in this setup process of $\mathcal{B}$ is to once again examine all its blocks and remove every block B such that either $S(\mathcal{A}_B) < 0$ or $\mathcal{A}_B$ is contained in $\mathcal{A}_{B'}$ for some other $B' \in \mathcal{B}$ for which $S(\mathcal{A}_B) \leq S(\mathcal{A}_{B'})$. The containment of one alignment in another is a notion completely analogous to that of block containment introduced above.

3.2 Alignments from Blocks

The $\mathcal{B}$ that we now have contains k-subsequence blocks exclusively, all having nonnegative-score alignments that are not contained in one another (except when the container has a lower score). In this new phase of the heuristic we build a weighted acyclic directed graph D from $\mathcal{B}$. Manipulating this graph appropriately eventually yields the desired alignment $\mathcal{A}$ of $s_1, \ldots, s_k$.

The node set of D is $\mathcal{B} \cup \{s, t\}$, where s and t are two special nodes. In D, an edge exists directed from s to each $B \in \mathcal{B}$, and also from each $B \in \mathcal{B}$ to t. No other edges are incident to s or t, which are then a source and a sink in D, respectively (i.e., s only has outgoing edges, t only incoming edges). The additional edges of D are deployed among the members of $\mathcal{B}$ in the following manner. For $B, B' \in \mathcal{B}$, an edge exists directed from B to B' if every subsequence of B starts to the left of the corresponding subsequence of B' in the appropriate sequence of $s_1, \ldots, s_k$. In addition, if B and B' overlap, then $\mathcal{A}_B$ and $\mathcal{A}_{B'}$ are also required to be identical in all the overlapping columns. Edges deployed in this manner lead, clearly, to an acyclic directed graph.

In D, both edges and nodes have weights. Edge weights depend on how the blocks intersect the sequences $s_1, \ldots, s_k$. Specifically, if an edge exists from B to B' and the two blocks are nonoverlapping, then its weight is $-x$, where x is the standard deviation of the intervening sequence-segment lengths. Edges outgoing from s or incoming to t are weighted in the trivially analogous manner.

Weights for edges between overlapping blocks and node weights are computed similarly to each other (except for s and t, whose weights are equal to 0). If x is the number of residues in node B, then its weight is $x/\sqrt{k}$. In the case of an edge between the overlapping B and B', we let x be the number of common residues and set the edge's weight to $-x/\sqrt{k}$. We remark, finally, that this weight-assignment methodology is very similar to the one in [4], the main difference being that we count residues instead of alignment sizes.

Having built D, we are then two further steps away from the final alignment $\mathcal{A}$. The first step is to find an s-to-t directed path in D whose weighted length is greatest. Since D is acyclic, this can be achieved efficiently. Every block B appearing on this optimal path immediately contributes $\mathcal{A}_B$ as part of $\mathcal{A}$, but there still remain unaligned sequence segments.

The second step and final action of the heuristic is then to complete the missing positions of $\mathcal{A}$. We describe what is done between nonoverlapping successive blocks, but clearly the same has to be applied to the left of the first block on the optimal path and to the right of the last block. Let B and B' be nonoverlapping blocks appearing in succession on the optimal path. Let $t_1, \ldots, t_k$ be the intervening subsequences of $s_1, \ldots, s_k$ that are still unaligned. We select the largest of $t_1, \ldots, t_k$ and use it to initialize a new alignment along with as many gap characters as needed for every one of $t_1, \ldots, t_k$ that is empty. We then visit each of the remaining subsequences in nonincreasing length order and align it to the current, partially built new alignment. The method used here is totally analogous to the one used in Section 3.1 for providing every block with exactly k subsequences, the only difference being that a global (as opposed to semi-global) procedure is used.

4 Computational Results

We have conducted extensive experimentation in order to evaluate the performance of the heuristic of Section 3. Our strategy has been to employ the BAliBASE suite [5] as the source of sequence sets, and to seek comparative results vis-à-vis some prominent approaches, namely CLUSTAL W [6], PRRN [7], DIALIGN [8, 9], T-COFFEE [10], and MAFFT [11]. The BAliBASE suite comprises 167 families of amino-acid sequences divided into eight reference sets, each of which especially constructed to emphasize some of the most common scenarios related to multiple sequence alignment. The suite contains a reference alignment for each of its families, in most cases along with motif annotations given the reference alignment. We have concentrated our experiments on the families for which such annotations are available, namely the first five reference sets.

The metrics we use to evaluate a certain alignment $\mathcal{A}$ are motif-constrained versions of those originally introduced along with the BAliBASE suite: the sum-of-pairs score (SPS), denoted by $SPS(\mathcal{A})$, and the column score (CS), denoted by $CS(\mathcal{A})$. If we let p_{abc} be a 0-1 variable such that $p_{abc} = 1$ if and only if the residues $\mathcal{A}[a, c]$ and $\mathcal{A}[b, c]$ share the same column in the BAliBASE reference alignment for the same sequences, then the version of $SPS(\mathcal{A})$ we use is the average value of p_{abc} over the residue pairs that are annotated as belonging to motifs in the reference alignment. Similarly, if C_c is the 0-1 variable having value 1 if and only if all the residues in column c of $\mathcal{A}$ also share a column in the reference alignment, then we use $CS(\mathcal{A})$ as the average value of C_c over the columns of motifs in the reference alignment.

All the experiments with the heuristic of Section 3 were carried out with $M = 1, \ldots, 4$, $L = 20$, and $N = 200$. Combining the numbers of optimal global and local pairwise alignments as indicated in the introduction to Section 3 was achieved via convex combinations with proportions that varied from one BAliBASE reference set to another.[1] The set covers we used are the one introduced in [12], here denoted by $\mathcal{I}$, and the one from [13], here denoted by $\mathcal{S}$. In both cases, R is the set of amino acids. We used the substitution matrices `BLOSUM62` [14], `PAM250` [15], and `VTML160` [16], together with gap costs as given in [17] for `BLOSUM62` and `PAM250` or as in [18] for `VTML160`.

We have found $M = 2$ to be the best choice nearly always, and found also that the `VTML160`-$\mathcal{S}$ pair for the combination of a substitution matrix and a set cover appears as a first choice most often, with the only noteworthy exception of Reference Set 4, in which case it seems best to use the pair `BLOSUM62`-$\mathcal{I}$. The results we present next refer to comparing the heuristic of Section 3 given these choices to the five competing approaches mentioned earlier. The approaches that predate the introduction of the BAliBASE suite were run with default parameters (this is the case of CLUSTAL W, PRRN, and DIALIGN), while the others, having appeared with computational results on the BAliBASE suite

[1] Following initial tuning experiments, weights for the numbers of optimal global pairwise alignments were as follows: 0.10 (Reference Set 1.1), 0.20 (Set 1.2), 0.05 (Set 1.3), 0.55 (Set 2), 0.50 (Set 3), 1.00 (Set 4), and 0.15 (Set 5).

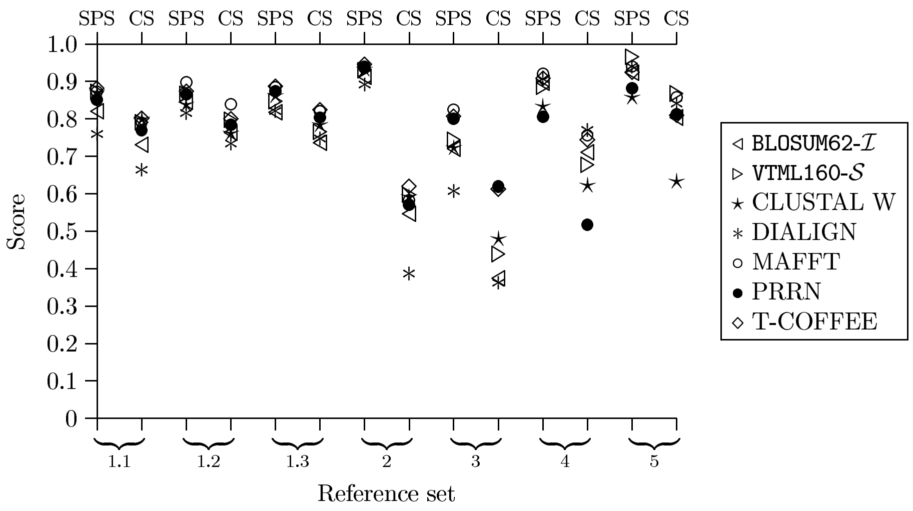


Fig. 2. Average scores for our best choices and the five competing approaches

when first published, were run with their best parameter choices (this applies to T-COFFEE and to MAFFT).

Comparative results are given in Figure 2, where we show average SPS and CS values inside each of the BAliBASE reference sets we considered. It is clear from Figure 2 that no absolute best can be identified throughout all the reference sets. As we examine the reference sets individually, though, we see that at least one of the two substitution-matrix, set-cover pairs used with our heuristic is in general competitive with the best contender. Noteworthy situations are the superior performance of our heuristic on Reference Set 5, and also its weak performance on Reference Set 3. As for the corresponding running times, our current implementation performs competitively as well when compared to the others: on an Intel Pentium 4 processor running at 1.8 GHz with 1 Gbytes of main memory, ours has taken from 1843.74 to 2010.81 seconds to complete, so it is slower than the fastest performer (MAFFT, 95.59 seconds) but faster than the slowest one (T-COFFEE, 4922.50 seconds).

5 Concluding Remarks

Many of our heuristic's details can still undergo improvements that go beyond the mere search for a more efficient implementation. One possibility is clearly the use of potentially better pairwise alignments, both global and local, when they are needed as described in Section 3. This possibility is already exploited by T-COFFEE, which not only employs a position-specific score matrix, but also uses CLUSTAL W to obtain global pairwise alignments, among other things. We also see improvement possibilities in the block- and alignment-extension methods described at the ends of Sections 3.1 and 3.2, respectively. In these two occasions, sequences or subsequences are considered in nonincreasing order of their lengths, which of course is an approach simple to the point of in no way taking into account the biological significance of the sequences or subsequences.

It is also apparent from our presentation of the heuristic in Section 3 that several options exist for many of its building parts. This refers not only to choosing parameter values but also to selecting auxiliary algorithms at several points. Whether better choices exist in terms of yielding even more significant alignments, and doing it perhaps faster as well, remains to be verified.

Acknowledgments

The authors acknowledge partial support from CNPq, CAPES, and a FAPERJ BBP grant.

References

1. Manthey, B.: Non-approximability of weighted multiple sequence alignment. Theoretical Computer Science **296** (2003) 179–192
2. Li, T.P., Fan, K., Wang, J., Wang, W.: Reduction of protein sequence complexity by residue grouping. Protein Engineering **16** (2003) 323–330
3. Gusfield, D.: Algorithms on Strings, Trees, and Sequences. Cambridge University Press, Cambridge, UK (1997)
4. Zhao, P., Jiang, T.: A heuristic algorithm for multiple sequence alignment based on blocks. Journal of Combinatorial Optimization **5** (2001) 95–115
5. Bahr, A., Thompson, J.D., Thierry, J.C., Poch, O.: BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. Nucleic Acids Research **29** (2001) 323–326
6. Thompson, J.D., Higgins, D.G., Gibson, T.J.: CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties, and weight matrix choice. Nucleic Acids Research **22** (1994) 4673–4680
7. Gotoh, O.: Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. Journal of Molecular Biology **264** (1996) 823–838
8. Morgenstern, B., Frech, K., Dress, A., Werner, T.: DIALIGN: finding local similarities by multiple sequence alignment. Bioinformatics **14** (1998) 290–294
9. Morgenstern, B.: DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics **15** (1999) 211–218
10. Notredame, C., Higgins, D.G., Heringa, J.: T-Coffee: a novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology **302** (2000) 205–217
11. Katoh, K., Misawa, K., Kuma, K., Miyata, T.: MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Research **30** (2002) 3059–3066
12. Ioerger, T.R.: The context-dependence of amino acid properties. In: Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology. (1997) 157–166

13. Smith, R.F., Smith, T.F.: Automatic generation of primary sequence patterns from sets of related protein sequences. Proceedings of the National Academy of Sciences USA **87** (1990) 118–122
14. Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences USA **89** (1992) 10915–10919
15. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. In Dayhoff, M.O., ed.: Atlas of Protein Sequence and Structure. Volume 5, supplement 3. National Biomedical Research Foundation, Washington, DC (1978) 345–352
16. Müller, T., Spang, R., Vingron, M.: Estimating amino acid substitution models: a comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method. Molecular Biology and Evolution **19** (2002) 8–13
17. Vogt, G., Etzold, T., Argos, P.: An assessment of amino acid exchange matrices in aligning protein sequences: the twilight zone revisited. Journal of Molecular Biology **249** (1995) 816–831
18. Green, R.E., Brenner, S.E.: Bootstrapping and normalization for enhanced evaluations of pairwise sequence comparison. Proceedings of the IEEE **90** (2002) 1834–1847

A Methodology for Determining Amino-Acid Substitution Matrices from Set Covers

Alexandre H.L. Porto and Valmir C. Barbosa[⋆]

Universidade Federal do Rio de Janeiro,
Programa de Engenharia de Sistemas e Computação, COPPE,
Caixa Postal 68511, Rio de Janeiro - RJ 21941-972, Brazil
valmir@cos.ufrj.br

Abstract. We introduce a new methodology for the determination of amino-acid substitution matrices for use in the alignment of proteins. The new methodology is based on a pre-existing set cover on the set of residues and on the undirected graph that describes residue exchangeability given the set cover. For fixed functional forms indicating how to obtain edge weights from the set cover and, after that, substitution-matrix elements from weighted distances on the graph, the resulting substitution matrix can be checked for performance against some known set of reference alignments and for given gap costs. Finding the appropriate functional forms and gap costs can then be formulated as an optimization problem that seeks to maximize the performance of the substitution matrix on the reference alignment set. We give computational results on the BAliBASE suite using a genetic algorithm for optimization. Initial results indicate that it is possible to obtain substitution matrices whose performance is either comparable to or surpasses that of several others.

Keywords: Sequence alignment; Substitution matrix; Residue set cover.

1 Introduction

One of the most central problems of computational molecular biology is to align two sequences of residues, a residue being generically understood as a nucleotide or an amino acid, depending respectively on whether the sequences under consideration are nucleic acids or proteins. This problem lies at the heart of several higher-level applications, such as heuristically searching sequence bases or aligning a larger number of sequences concomitantly for the identification of special common substructures (the so-called motifs, cf. [1]) that encode structural or functional similarities of the sequences or yet the sequences' promoter regions in the case of nucleic acids, for example.

Finding the best alignment between two sequences is based on maximizing a scoring function that quantifies the overall similarity between the sequences. Normally this similarity function has two main components. The first one is a symmetric matrix, known as the substitution matrix for the set of residues

[⋆] Corresponding author.

under consideration, which gives the contribution the function is to incur when two residues are aligned to each other. The second component represents the cost of aligning a residue in a sequence to a gap in the other, and gives the negative contribution to be incurred by the similarity function when this happens. There is no consensually accepted, general-purpose criterion for selecting a substitution matrix or a gap-cost function. Common criteria here include those that stem from structural or physicochemical characteristics of the residues and those that somehow seek to reproduce well-known alignments as faithfully as possible [2].

We then see that, even though an optimal alignment between two sequences is algorithmically well understood and amenable to being computed efficiently, the inherent difficulty of selecting appropriate scoring parameters suggests that the problem is still challenging in a number of ways. This is especially true of the case of protein alignment, owing primarily to the fact that the set of residues is significantly larger than in the case of nucleic acids, and also to the existence of a multitude of criteria whereby amino acids can be structurally or functionally exchanged by one another.

For a given structural or physicochemical property (or set of properties) of amino acids, this exchangeability may be expressed by a set cover of the set of all amino acids, that is, by a collection of subsets of that set that includes every amino acid in at least one subset. Each of these subsets represents the possibility of exchanging any of its amino acids by any other. Set covers in this context have been studied extensively [3] and constitute our departing point in this paper. As we describe in Section 2, we introduce a new methodology for discovering both an appropriate substitution matrix and gap-cost parameters that starts by considering an amino-acid set cover. It then builds a graph from the set cover and sets up an optimization problem whose solution is the desired substitution matrix and gap costs.

The resulting optimization problem is defined on a set of target sequence pairs, preferably one that embodies as great a variety of situations as possible. The target pairs are assumed to have known alignments, so the optimal solution to the problem of finding parameters comprises the substitution matrix and the gap costs whose use in a predefined alignment algorithm yields alignments of the target pairs that in some sense come nearest the known alignments of the same pairs. Our optimization problem is set up as a problem of combinatorial search, being therefore highly unstructured and devoid of any facilitating differentiability properties. Reasonable ways to approach its solution are then all heuristic in nature. In Section 3, we present the results of extensive computational experiments that employ an evolutionary algorithm and targets the BAliBASE pairs of amino-acid sequences [4].

Notice, in the context of the methodology categorization we mentioned earlier in passing, that our new methodology is of a dual character: it both relies on structural and physicochemical similarities among amino acids and depends on a given set of aligned sequences in order to arrive at a substitution matrix and gap costs.

We close in Section 4 with conclusions.

2 The Methodology

We describe our methodology for sequences on a generic set R of residues and only specialize it to the case of proteins in Section 3. Given two residue sequences X and Y of lengths x and y, respectively, a global alignment of X and Y can be expressed by the $2 \times z$ matrix A having the property that its first line, when read from left to right, is X possibly augmented by interspersed gaps, the same holding for the second line and Y, so long as no column of A comprises gaps only. It follows that $\max\{x, y\} \leq z \leq x + y$. In the case of a local alignment, that is, an alignment of a subsequence of X and another of Y, this matrix representation remains essentially unchanged, provided of course that x and y are set to indicate the sizes of the two subsequences.

For a given substitution matrix S and a pair (h, g) of gap costs,[1] the similarity score of alignment A, denoted by $F_S^{h,g}(A)$, is given by

$$F_S^{h,g}(A) = \sum_{j=1}^{z} f_S^{h,g}(A(1,j), A(2,j)), \tag{1}$$

where $f_S^{h,g}(A(1,j), A(2,j))$ gives the contribution of aligning $A(1,j)$ to $A(2,j)$ as either $S(A(1,j), A(2,j))$, if neither $A(1,j)$ nor $A(2,j)$ is a gap; or $-(h + g)$, if either $A(1,j)$ or $A(2,j)$ is the first gap in a contiguous group of gaps; or yet $-g$, if either $A(1,j)$ or $A(2,j)$ is the kth gap in a contiguous group of gaps for $k > 1$. An optimal global alignment of X and Y is one that maximizes the similarity score of (1) over all possible global alignments of the two sequences. An optimal local alignment of X and Y, in turn, is the optimal global alignment of the subsequences of X and Y for which the similarity score is maximum over all pairs of subsequences of the two sequences. The set of all optimal alignments of X and Y may be exponentially large in x and y, but it does nonetheless admit a concise representation as a matrix or directed graph that can be computed efficiently by well-known dynamic programming techniques, regardless of whether a global alignment of the two sequences is desired or a local one. We refer to this representation as $\mathcal{A}^*_{X,Y}$.

Our strategy for the determination of a suitable substitution matrix starts with a set cover $\mathcal{C} = \{C_1, \ldots, C_c\}$ of the residue set R, that is, $\mathcal{C}$ is such that $C_1 \cup \cdots \cup C_c = R$. Next we define G to be an undirected graph of node set R having an edge between two nodes (residues) u and v if and only if at least one of $C_1, \ldots, C_c$ contains both u and v. Graph G provides a natural association between how exchangeable a node is by another and the distance between them in the graph. Intuitively, the closer two nodes are to each other in G the more exchangeable they are and we expect an alignment of the two to contribute relatively more positively to the overall similarity score. Quantifying this intuition involves crucial decisions, so we approach the problem in two careful steps, each

[1] For $k > 0$, we assume the customary affine function $p(k) = h + gk$ with $h, g > 0$ to express the cost of aligning the kth gap of a contiguous group of gaps in a line of A to a residue in the other line as $p(k) - p(k-1)$, assuming $p(0) = 0$ [5].

leaving considerable room for flexibility. The first step consists of turning G into a weighted graph, that is, assigning nonnegative weights to its edges, and then computing the weighted distance between all pairs of nodes. The second step addresses the turning of these weighted distances into elements of a substitution matrix so that larger distances signify ever more restricted exchangeability.

Let us begin with the first step. For (u, v) an edge of G, let $w(u, v)$ denote its weight. We define the value of $w(u, v)$ on the premise that, if the exchangeability of u and v comes from their concomitant membership in a large set of $\mathcal{C}$, then it should eventually result in a smaller contribution to the overall similarity score than if they were members of a smaller set. In other words, the former situation bespeaks an intuitive “weakness” of the property that makes the two residues exchangeable. In broad terms, then, we should let $w(u, v)$ be determined by the smallest of the sets of $\mathcal{C}$ to which both u and v belong, and should also let it be a nondecreasing function of the size of this smallest set.

Let c^- be the size of the smallest set of $\mathcal{C}$ and c^+ the size of its largest set. Let $c^-_{u,v}$ be the size of the smallest set of $\mathcal{C}$ of which both u and v are members. We consider two functional forms according to which $w(u, v)$ may depend on $c^-_{u,v}$ as a nondecreasing function. Both forms force $w(u, v)$ to be constrained within the interval $[w^-, w^+]$ with $w^- \geq 0$. For $\lambda \geq 1$, the first form is the convex function

$$w_1(u, v) = w^- + (w^+ - w^-) \left(\frac{c^-_{u,v} - c^-}{c^+ - c^-} \right)^{\lambda}, \tag{2}$$

while the second is the concave function

$$w_2(u, v) = w^+ - (w^+ - w^-) \left(\frac{c^+ - c^-_{u,v}}{c^+ - c^-} \right)^{\lambda}. \tag{3}$$

Having established weights for all the edges of G, let $d_{u,v}$ denote the weighted distance between nodes u and v. Clearly, $d_{u,u} = 0$ and, if no path exists in G between u and v (i.e., G is not connected and the two nodes belong to two different connected components), then $d_{u,v} = \infty$.

Carrying out the second step, which is obtaining the elements of the substitution matrix from the weighted distances on G, involves difficult choices as well. While, intuitively, it is clear that residues separated by larger weighted distances in G are to be less exchangeable for each other than residues that are closer to each other (in weighted terms) in G, the functional form that the transformation of weighted distances into substitution-matrix elements is to take is once again subject to somewhat arbitrary decisions. What we do is to set $S(u, v) = 0$ if $d_{u,v} = \infty$, and to consider two candidate functional forms for the transformation in the case of finite distances.

Let us initially set $[S^-, S^+]$ as the interval within which each element of the substitution matrix S is to be constrained (we assume $S^- > 0$ for consistency with the substitution-matrix element that goes with an infinite distance, whose value we have just set to 0). Let us also denote by d^+ the largest (finite) weighted distance occurring in G for the choice of weights at hand. We then consider two functional forms for expressing the dependency of $S(u,v)$, as a nonincreasing function, upon a finite $d_{u,v}$. For $\mu \geq 1$, we once again consider a convex function,

$$S_1(u,v) = S^- + (S^+ - S^-)\left(\frac{d^+ - d_{u,v}}{d^+}\right)^{\mu}, \tag{4}$$

and a concave one,

$$S_2(u,v) = S^+ - (S^+ - S^-)\left(\frac{d_{u,v}}{d^+}\right)^{\mu}. \tag{5}$$

Once we decide on one of the two functional forms (2) or (3), and similarly on one of (4) or (5), and also choose values for w^-, w^+, λ, S^-, S^+, and μ, then the substitution matrix S as obtained from $\mathcal{C}$ is well-defined and, together with the gap-cost parameters h and g, can be used to find the representation $\mathcal{A}^*_{X,Y}$ of the set of all optimal (global or local) alignments between the two sequences X and Y. The quality of our choices regarding functional forms and parameters, and hence the quality of the resulting S, h, and g, can be assessed if a reference alignment, call it $A^r_{X,Y}$, is available for the two sequences. When this is the case, we let $\rho_S^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ be the fraction of the columns of $A^r_{X,Y}$ that also appear in at least one of the alignments that are represented in $\mathcal{A}^*_{X,Y}$. The substitution matrix S, and also h and g, are then taken to be as good for $A^r_{X,Y}$ as $\rho_S^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ is close to 1.

Thus, given a residue set cover $\mathcal{C}$ and a set $\mathcal{A}^r$ of reference alignments (each alignment on a different pair of sequences over the same residue set R), obtaining the best possible substitution matrix S and gap-cost parameters h and g can be formulated as the following optimization problem: find functional forms and parameters that maximize some (for now unspecified) average of $\rho_S^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ over all pairs (X,Y) of sequences such that $A^r_{X,Y} \in \mathcal{A}^r$. In the next section, we make this definition precise when residues are amino acids and proceed to the description of computational results.

3 Computational Results

Let b_w be a two-valued variable indicating which of (2) or (3) is to be taken as the functional form for the edge weights, and similarly let b_S indicate which of (4) or (5) is to give the functional form for the elements of S. These new parameters defined, we begin by establishing bounds on the domains from which each of the other eight parameters involved in the optimization problem may take values, and also make those domains discrete inside such bounds by taking equally spaced delimiters. For the purposes of our study in this section, this results in what is shown in Table 1.

The parameter domains shown in Table 1 make up for over 3.7 trillion possible combinations, yielding about 1.6 billion different substitution matrices. The set of all such combinations seems to be structured in no usable way, so finding the best combination with respect to some set of reference alignments as discussed in

Table 1. Parameters and their domains

Parameter	Description	Domain
b_w	Selects between (2) and (3)	$\{1, 2\}$
w^-	Least possible edge weight	$\{0.5, 0.55, \ldots, 1\}$
w^+	Greatest possible edge weight	$\{1, 1.125, \ldots, 5\}$
λ	Exponent for use in (2) or (3)	$\{1, 1.125, \ldots, 5\}$
b_S	Selects between (4) and (5)	$\{1, 2\}$
S^-	Least possible element of S	$\{0.5, 0.55, \ldots, 1\}$
S^+	Greatest possible element of S	$\{1, 1.25, \ldots, 25\}$
μ	Exponent for use in (4) or (5)	$\{1, 1.125, \ldots, 5\}$
h	Initialization gap cost	$\{2, 2.5, \ldots, 30\}$
g	Extension gap cost	$\{0.25, 0.375, \ldots, 5\}$

Section 2 must not depend on any technique of explicit enumeration but rather on some heuristic approach.

The approach we use in this section is to employ an evolutionary algorithm for finding the best possible combination within reasonable time bounds. Each individual for this algorithm is a 10-tuple indicating one of the possible combination of parameter values. Our evolutionary algorithm is a standard generational genetic algorithm. It produces a sequence of 100-individual generations, the first of which is obtained by randomly choosing a value for each of the 10 parameters in order to produce each of its individuals. Each of the subsequent generations is obtained from the current generation by a combination of crossover and mutation operations, following an initial elitist step whereby the 5 fittest individuals of the current generation are copied to the new one.

While the new generation is not full, either a pair of individuals is selected from the current generation to undergo crossover (with probability 0.5) or one individual is selected to undergo a single-locus mutation (with probability 0.5).[2] The pair of individuals resulting from the crossover, or the single mutated individual, is added to the new generation, unless an individual that is being added is identical to an individual that already exists in the population. When this happens, the duplicating individual is substituted for by a randomly generated individual. Selection is performed in proportion to the individuals' linearly normalized fitnesses.[3]

The crux of this genetic algorithm is of course how to assess an individual's fitness, and this is where an extant set of reference alignments $\mathcal{A}^r$ comes in. In our

[2] Both the crossover point and the locus for mutation are chosen at random, essentially with the parameters' domains in mind, so that the probability that such a choice singles out a parameter whose domain has size a is proportional to $\log a$. Mutating the parameter's value is achieved straightforwardly, while breaking the 10-tuples for crossover requires the further step of interpreting the parameter as a binary number.

[3] This means that, for $1 \leq k \leq 100$, the kth fittest individual in the generation is selected with probability proportional to $L - (L - 1)(k - 1)/99$, where L is chosen so that the expression yields a value L times larger for the fittest individual than it does for the least fit (for which it yields value 1). We use $L = 10$ throughout.

study we take $\mathcal{A}^r$ to be the set of alignments present in the BAliBASE suite. It contains 167 families of amino-acid sequences arranged into eight reference sets. For each family of the first five reference sets two pieces of reference information are provided: a multiple alignment of all the sequences in the family and a demarcation of the relevant motifs given the multiple alignment. Families in the remaining three reference sets are not provided with motif demarcations, so we refrain from using them in our experiments, since the fitness function that we use relies on reference motifs as well. Note that, even though the BAliBASE suite is targeted at multiple sequence alignments, each such alignment trivially implies a pairwise alignment for all sequence pairs in each family and also motif fragments for each pair. Our set $\mathcal{A}^r$ then comprises every sequence pair from the BAliBASE suite for which a reference alignment exists with accompanying motif demarcation.

The organization of the BAliBASE suite suggests a host of possibilities for evaluating the efficacy of a substitution matrix S and of gap-cost parameters h and g. For a pair of sequences (X, Y), whose reference alignment is $A^r_{X,Y} \in \mathcal{A}^r$, and recalling that $\mathcal{A}^*_{X,Y}$ represents the set of all optimal alignments of X and Y given S, h, and g, we use four variants of the $\rho_S^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ of Section 2 as the bases of the fitness function to be used by the genetic algorithm. These are denoted by $\rho_{S,1}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ through $\rho_{S,4}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ and differ among themselves as to which of the columns of the reference alignment are checked to be present in at least one of the optimal alignments. We let them be as follows:

- $\rho_{S,1}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ is based on all the columns of $A^r_{X,Y}$;
- $\rho_{S,2}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ is based on all the columns of $A^r_{X,Y}$ that contain no gaps;
- $\rho_{S,3}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ is based on all the columns of $A^r_{X,Y}$ that lie within motifs;
- $\rho_{S,4}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ is based on all the columns of $A^r_{X,Y}$ that lie within motifs and contain no gaps.

These defined, we first average each one of them over $\mathcal{A}^r$ before combining them into a fitness function. The average that we take is computed in the indirectly weighted style of [6], which aims at preventing any family with overly many pairs, or any pair on which S, h, and g are particularly effective, from influencing the average too strongly.

The weighting takes place on an array having 10 lines, one for each of the nonoverlapping 0.1-wide intervals within $[0, 1]$, and one column for each of the BAliBASE families. Initially each pair (X, Y) having a reference alignment $A^r_{X,Y}$ in $\mathcal{A}^r$ is associated with the array cell whose column corresponds to its family and whose line is given by the interval within which the identity score of the reference alignment $A^r_{X,Y}$ falls. This score is the ratio of the number of columns of $A^r_{X,Y}$ whose two amino acids are identical to the number of columns that have no gaps (when averaging $\rho_{S,3}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ or $\rho_{S,4}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$, only columns that lie within motifs are taken into account).

For $1 \leq k \leq 4$, we then let $\rho_{S,k}^{h,g}(\mathcal{A}^r)$ be the average of $\rho_{S,k}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ over $\mathcal{A}^r$ obtained as follows. First take the average of $\rho_{S,k}^{h,g}(A^r_{X,Y}, \mathcal{A}^*_{X,Y})$ for each array cell over the sequence pairs (X, Y) that are associated with it (cells with no pairs are ignored). Then $\rho_{S,k}^{h,g}(\mathcal{A}^r)$ is computed by first averaging those averages that correspond to the same line of the array and finally averaging the resulting numbers (note that lines whose cells were all ignored for having no sequence pairs associated with them do not participate in this final average).

We are then in position to state the definition of our fitness function. We denote it by $\varphi_S^{h,g}(\mathcal{A}^r)$ to emphasize its dependency on how well S, h, and g lead to alignments that are in good accord with the alignments of $\mathcal{A}^r$. It is given by the standard Euclidean norm of the four-dimensional vector whose kth component is $\rho_{S,k}^{h,g}(\mathcal{A}^r)$, that is,

$$\varphi_S^{h,g}(\mathcal{A}^r) = \sqrt{\left[\rho_{S,1}^{h,g}(\mathcal{A}^r)\right]^2 + \cdots + \left[\rho_{S,4}^{h,g}(\mathcal{A}^r)\right]^2}. \tag{6}$$

Clearly, $0 \leq \varphi_S^{h,g}(\mathcal{A}^r) \leq 2$ always.

We found through initial experiments that carrying over with our algorithm for each single generation requires roughly 13 to 14 hours on an Intel Pentium 4 processor running at 2.26 GHz and equipped with 512 Mbytes of main memory. Practically all of this effort is related to computing $\varphi_S^{h,g}(\mathcal{A}^r)$ for each individual in the current population, and because this is done in a manner that is fully independent from any other individual, we can speed the overall computation up nearly optimally by simply bringing more processors into the effort.

The results we describe next were obtained on four processors running in parallel and for the following simplifications. We concentrated solely on evolving individuals under global alignments for the set cover of [7], and considered, in addition, only the subset of $\mathcal{A}^r$, denoted by $\mathcal{A}^{r,1}$, comprising sequence pairs that are relative to the BAliBASE reference set 1. In this case, the fitness function to be maximized is $\varphi_S^{h,g}(\mathcal{A}^{r,1})$, defined as in (6) when $\mathcal{A}^{r,1}$ substitutes for $\mathcal{A}^r$. Given these simplifications, computing through each generation has taken roughly 20 minutes.

The substitution matrices we have used for the sake of comparison are `BC0030` [8], `BENNER74` [9], `BLOSUM62` [10], `FENG` [11], `GONNET` [12], `MCLACH` [13], `NWSGAPPEP` [14], `PAM250` [15], `RAO` [16], `RUSSELL-RH` [17], and `VTML160` [18]. The gap-cost parameters h and g we used with them are the ones from [8] for `BC0030` and `RUSSELL-RH`, from [19] for `VTML160`, and from [6] for all others.

Our results are summarized in the four plots of Figure 1, each giving the evolution of one of $\rho_{S,1}^{h,g}(\mathcal{A}^{r,1}), \ldots, \rho_{S,4}^{h,g}(\mathcal{A}^{r,1})$ for the fittest individuals along the generations. We show this evolution against dashed lines that delimit the intervals within which the corresponding figures for the matrices used for comparison are located. Clearly, the genetic algorithm very quickly produces a substitution matrix, with associated gap costs, that surpasses this interval as far as the fitness components $\rho_{S,3}^{h,g}(\mathcal{A}^{r,1})$ and $\rho_{S,4}^{h,g}(\mathcal{A}^{r,1})$ are concerned, even though it lags behind in terms of $\rho_{S,1}^{h,g}(\mathcal{A}^{r,1})$ and $\rho_{S,2}^{h,g}(\mathcal{A}^{r,1})$. This substitution matrix, it turns out, is then superior to all those other matrices when it comes to stressing alignment columns that lie within motifs.

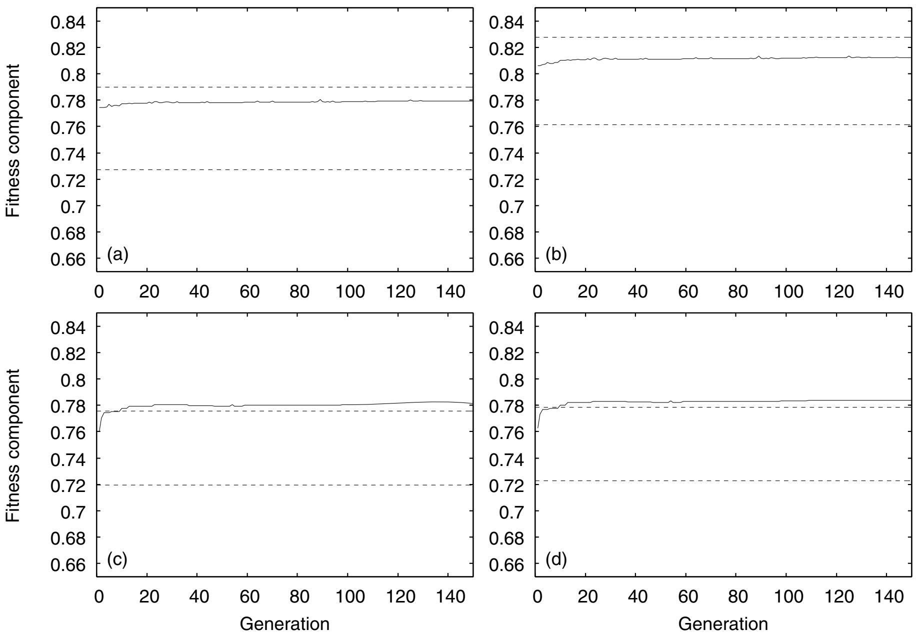


Fig. 1. Evolution of the fitness components $\rho^{h,g}_{S,1}(\mathcal{A}^{r,1}), \ldots, \rho^{h,g}_{S,4}(\mathcal{A}^{r,1})$, shown respectively in (a) through (d)

4 Concluding Remarks

We have introduced a new methodology for the determination of amino-acid substitution matrices. The new methodology starts with a set cover of the residue alphabet under consideration and builds an undirected graph in which node vicinity is taken to represent residue exchangeability. The desired substitution matrix arises as a function of weighted distances in this graph. Determining the edge weights, and also how to convert the resulting weighted distances into substitution-matrix elements, constitute the outcome of an optimization process that runs on a set of reference sequence alignments and also outputs gap costs for use with the substitution matrix. Our methodology is then of a hybrid nature: it relies both on the structural and physicochemical properties that underlie the set cover in use and on an extant set of reference sequence alignments.

The optimization problem to be solved is well-defined: given parameterized functional forms for turning cover sets into edge weights and weighted distances into substitution-matrix elements, the problem asks for parameter values and gap costs that maximize a certain objective function on the reference set of alignments. We have reported on computational experiments that use a genetic algorithm as optimization method and the BAliBASE suite as the source of the required reference alignments.

Our results so far indicate that the new methodology is capable of producing substitution matrices whose performance falls within the same range of a number of known matrices' even before any optimization is actually performed (i.e., based on the random parameter instantiation that precedes the genetic algorithm); this alone, we believe, singles out our methodology as a principled way of determining substitution matrices that concentrates all the effort related to the structure and physicochemical properties of amino acids on the discovery of an appropriate set cover. They also indicate, in a restricted setting, that the methodology can yield substitution matrices that surpass all the others against which they were tested.

We have also found that strengthening this latter conclusion so that it holds in a wider variety of scenarios depends on how efficiently we can run the genetic algorithm. Fortunately, it appears that it is all a matter of how many processors can be amassed for the effort, since the genetic procedure is inherently amenable to parallel processing and highly scalable, too. There is, of course, also the issue of investigating alternative functional forms and parameter ranges to set up the optimization problem, and in fact the issue of considering other objective functions as well. Together with the search for faster optimization, these issues make for a very rich array of possibilities for further study.

Acknowledgments

The authors acknowledge partial support from CNPq, CAPES, and a FAPERJ BBP grant.

References

1. Sagot, M.F., Wakabayashi, Y.: Pattern inference under many guises. In Reed, B.A., Sales, C.L., eds.: Recent Advances in Algorithms and Combinatorics. Springer-Verlag, New York, NY (2003) 245–287
2. Valdar, W.S.J.: Scoring residue conservation. Proteins: Structure, Function, and Genetics **48** (2002) 227–241
3. Li, T.P., Fan, K., Wang, J., Wang, W.: Reduction of protein sequence complexity by residue grouping. Protein Engineering **16** (2003) 323–330
4. Bahr, A., Thompson, J.D., Thierry, J.C., Poch, O.: BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. Nucleic Acids Research **29** (2001) 323–326
5. Setubal, J., Meidanis, J.: Introduction to Computational Molecular Biology. PWS Publishing Company, Boston, MA (1997)
6. Vogt, G., Etzold, T., Argos, P.: An assessment of amino acid exchange matrices in aligning protein sequences: the twilight zone revisited. Journal of Molecular Biology **249** (1995) 816–831
7. Smith, R.F., Smith, T.F.: Automatic generation of primary sequence patterns from sets of related protein sequences. Proceedings of the National Academy of Sciences USA **87** (1990) 118–122
8. Blake, J.D., Cohen, F.E.: Pairwise sequence alignment below the twilight zone. Journal of Molecular Biology **307** (2001) 721–735

9. Benner, S.A., Cohen, M.A., Gonnet, G.H.: Amino acid substitution during functionally constrained divergent evolution of protein sequences. Protein Engineering **7** (1994) 1323–1332
10. Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences USA **89** (1992) 10915–10919
11. Feng, D.F., Johnson, M.S., Doolittle, R.F.: Aligning amino acid sequences: comparison of commonly used methods. Journal of Molecular Evolution **21** (1985) 112–125
12. Gonnet, G.H., Cohen, M.A., Benner, S.A.: Exhaustive matching of the entire protein sequence database. Science **256** (1992) 1443–1445
13. McLachlan, A.D.: Tests for comparing related amino-acid sequences. Cytochrome *c* and cytochrome *c*551. Journal of Molecular Biology **61** (1971) 409–424
14. Gribskov, M., Burgess, R.R.: Sigma factors from E. coli, B. subtilis, phage SP01, and phage T4 are homologous proteins. Nucleic Acids Research **14** (1986) 6745–6763
15. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. In Dayhoff, M.O., ed.: Atlas of Protein Sequence and Structure. Volume 5, supplement 3. National Biomedical Research Foundation, Washington, DC (1978) 345–352
16. Rao, J.K.M.: New scoring matrix for amino acid residue exchanges based on residue characteristic physical parameters. International Journal of Peptide and Protein Research **29** (1987) 276–281
17. Russell, R.B., Saqi, M.A.S., Sayle, R.A., Bates, P.A., Sternberg, M.J.E.: Recognition of analogous and homologous protein folds: analysis of sequence and structure conservation. Journal of Molecular Biology **269** (1997) 423–439
18. Müller, T., Spang, R., Vingron, M.: Estimating amino acid substitution models: a comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method. Molecular Biology and Evolution **19** (2002) 8–13
19. Green, R.E., Brenner, S.E.: Bootstrapping and normalization for enhanced evaluations of pairwise sequence comparison. Proceedings of the IEEE **90** (2002) 1834–1847

Multi-Objective Evolutionary Algorithm for Discovering Peptide Binding Motifs

Menaka Rajapakse[1,2], Bertil Schmidt[2], and Vladimir Brusic[3]

[1] Institute for Infocomm Research, 21 Heng Mui Keng Terrace, 119613, Singapore
menaka@i2r.a-star.edu.sg
[2] School of Computer Engineering, Nanyang Technological University, Block N4, Nanyang Avenue, 639798, Singapore
asbschmidt@ntu.edu.sg
[3] Australian Centre for Plant Functional Genomics, School of Land and Food Sciences and Institute for Molecular Bioscience, University of Queensland, Brisbane QLD 4072, Australia
v.brusic@uq.edu.au

Abstract. Multi-Objective Evolutionary Algorithms (MOEA) use Genetic Algorithms (GA) to find a set of potential solutions, which are reached by compromising trade-offs between the multiple objectives. This paper presents a novel approach using MOEA to search for a motif which can unravel rules governing peptide binding to medically important receptors with applications to drugs and vaccines target discovery. However, the degeneracy of motifs due to the varying physicochemical properties at the binding sites across large number of active peptides poses a challenge for the detection of motifs of specific molecules such as MHC Class II molecule I-A^{g7} of the non-obese diabetic (NOD) mouse. Several motifs have been experimentally derived for I-A^{g7} molecule, but they differ from each other significantly. We have formulated the problem of finding a consensus motif for I-A^{g7} by using MOEA as an outcome that satisfies two objectives: extract prior information by minimizing the distance between the experimentally derived motifs and the resulting matrix by MOEA; minimize the overall number of false positives and negatives resulting by using the putative MOEA-derived motif. The MOEA results in a Pareto optimal set of motifs from which the best motif is chosen by the Area under the Receiver Operator Characteristics (A_{ROC}) performance on an independent test dataset. We compared the MOEA-derived motif with the experimentally derived motifs and motifs derived by computational techniques such as MEME, RANKPEP, and Gibbs Motif Sampler. The overall predictive performance of the MOEA derived motif is comparable or better than the experimentally derived motifs and is better than the computationally derived motifs.

1 Introduction

Multi-Objective Evolutionary Algorithms (MOEA) have been effectively used in various domains to solve real-world complex search problems. Multi-objective problems need simultaneous optimization of number of competing objectives and result in a set of solutions called Pareto optimal set. These can be solved as a single objective optimization problem by combining all objectives into a single objective. Solving multi-objective problems as a single objective problem requires many simulation runs before achieving a set of Pareto optimal solutions. Another way of optimizing multi-objective problems is to use MOEAs that are able to find Pareto-optimal solutions in a single simulation run. A number of MOEAs has been suggested in the literature [1][2][3][4]. In a single objective GA a fitness function is used to asses the solution population, while MOEAs measure the dominance of individuals in the population and rank the solutions. A solution is said to be dominant if it is as good as or better than all the other solutions for all objectives [1]. Often, there is more than one dominant solution, which is known as the Pareto-front, representing the best solutions from a single iteration. A recently introduced MOEA known as non-dominated sorting genetic algorithm II (NSGA-II) by Deb *et al.* has been effectively applied for many real-world problems and has shown to be computationally efficient due to the incorporation of two new mechanisms: elitism and diversity-preservation [5]. Our study of motif discovery by using MOEA was carried out with NSGA-II that allows using multiple constraints [1].

In this paper, we describe the use of MOEAs in discovering peptide motifs associated with binding to I-A^{g7} molecule, which is involved in insulin-dependent diabetes mellitus (IDDM) in the non obese diabetic (NOD) mice [6-16]. A consensus motif that can describe peptide binding interactions to NOD mouse I-A^{g7} molecule has not been accurately defined by experimental or by computational means [25]. The motifs derived for this MHC Class II molecule are purely by experimental methods and majority of these experimentally derived motifs are mutually inconsistent thereby making it difficult to generalize across available datasets [26]. Moreover, no comparative study has been carried out to decipher the composition of amino acids in these binding peptides using computational methods.

A peptide motif is a representation of a conserved region of protein sequences that is linked to a specific biological function, for example peptide binding to a particular receptor. Multiple binding peptides normally contain a consensus motif that defines the binding rules to a particular receptor. A widely used representation of a motif is the quantitative matrix that contains k×20 coefficients where k corresponds to the length of the motif and 20 to the number of different amino acids in a protein sequence. The score for the prediction to a binder is calculated by summing or multiplying the matrix coefficients. Several experimental approaches have been attempted in finding motifs [6-12]. Popular computational tools available for finding motifs in protein sequences are: MEME [21], Gibbs motif sampler [22] and Rankpep [23].

In this study, a motif is expressed as a quantitative scoring matrix representing many possible interactions among amino acids, and the positions of binding to peptides. To derive a MOEA solution, two objectives were defined and the validity of the objectives is controlled by two constraints imposes on the effects of the objectives under the evaluation.

2 Materials and Methods

The motif discovery for I-A^{g7} dataset is performed with MOEA (NSGA-II). The approach for the discovery of a consensus motif is described in the subsequent sections.

2.1 I-A^{g7} Datasets

Ten different I-A^{g7} datasets were extracted from several independent studies [9-18]. Each experimental dataset provides peptides which are classified as binders or non-binders with a label which have been assigned according to their experimental binding affinities. The dataset available can be expressed as $\mathbf{D} = \{D_i: i=1,2,\ldots.d\}$ where d denotes the number of datasets. Let i th data set, $D_i = \{(\mathbf{x}_{ij}, v_{ij}): j = 1, 2,\ldots., n_i\}$ where $\mathbf{x}_{ij}$ is the jth sequence in the i th dataset and the label $v_{ij} \in \{\text{b}, \text{nb}\}$ indicates whether the sequence $\mathbf{x}_{ij}$ is a binder (b), or a non-binder (nb).

2.2 Experimental Motifs

Seven different reported k-mer motifs (MHC binging motifs are of k=9 amino acid length) [6-12] have been derived for predicting peptide binders to I-A^{g7} molecule. The residues, which contribute significantly to peptide binding, are called primary anchor residues and the positions they occur are called anchor positions. Anchor positions may be occupied by so called preferred residues which are tolerated, but alone contribute little to peptide binding strength. In [6-12] each motif describes amino acids at primary and secondary anchor positions, as well as “forbidden” amino acids at specific positions. We interpret these as anchor, tolerated, and non-tolerated amino acids. These experimental motifs can be expressed as $\mathbf{R} = \{R_i: i=1,2,\ldots.r\}$ where r denotes the number of experimentally found motifs. The Table 1 below illustrates an example of an experimentally derived Reizis peptide binding motif to I-A^{g7}.

Table 1. An illustration of a representative peptide binding motif to I-A^{g7} molecule (Reizis motif, [9]). The positions P4, P6 and P9 are primary anchor positions where binding is highly likely to occur.

Position	Anchor	Tolerated	Not tolerated
P1	VEQMHLPD	-	R
P2	-	-	-
P3	-	-	-
P4	**ILPV**	**HY**	**QEK**
P5	-	-	-
P6	**ATSNV**	**-**	**LYQK**
P7	QVYLHINRF	-	-
P8	-	-	-
P9	**ED**	**SM**	**LYTQK**

2.3 Binding Score Matrix Representation of a Motif

In this section we give a formal definition of the target model as a quantitative matrix. A k-mer motif in an amino acid sequence is usually characterized by a binding score matrix $\mathbf{Q} = \{q_{\text{ia}}\}_{\text{kx20}}$ where q_{ia} denotes the binding score of the site i of the motif when it occupies by the amino-acid $a \in \sum$ where $\sum$ denotes the set of 20 amino-acid residues. A binding score computed by adding the scores assigned for each amino acid in the respective positions of a *k*-mer motif not only indicates the likelihood of the presence of a particular motif but also determines the likelihood that a sequence containing the motif that binds to another sequence. Therefore, a binding score matrix can be viewed as a quantification of a real biological functioning or binding of the motif to other peptides. Given a binding score matrix **Q** of size $k\times 20$ we define the *binding score, s* for a *k*–mer motif, m^*, starting at the position j^* in a sequence $x=(x_1, x_2 \cdots x_n)$ of length *n* as:

$$s = \max_{j\in\{1,\ldots,n-k+1\}} s_j \tag{1}$$

$$\text{where } s_j = \sum_{i=0,\ldots,k-1} q_{ix_{i+j}} \tag{2}$$

And the location of the motif is given by

$$j^* = \arg\max_{j\in\{1,\cdots,n-k+1\}} s_j \tag{3}$$

We denote $s(x_{j*} : m^*)$ to indicate the binding score of the motif m^* present at j^* position of the peptide sequence x.

2.4 Multi-Objective Evolutionary Algorithm: NSGA-II

NSGA-II incorporates several mechanisms that facilitate faster and better convergence of a solution population for multi-objective problems. These mechanisms include non-dominated sorting, elitism, diversity preservation, and constrained handling. A solution is said to be dominant if it is as good as or better than the other solutions with respect to all objectives. Non-dominated sorting refers to sorting of individuals that are not dominated by any other individual in the population with respect to every objective. All the non-dominated individuals are assigned the same fitness value. The same procedure is then carried out on the remaining population until a new set of non-dominated solutions is found. The solutions found in the subsequent rounds are assigned a fitness value lower than in the previous round and the process continues on until the whole population is partitioned into non-dominated fronts with diverse fitness values. The elitism prevents loosing fit individuals encountered in earlier generations by allowing parents to compete with offspring. In NSGA-II, the diversity of Pareto-optimal solutions is maintained by imposing a measure known as crowding distance measure. More details on these mechanisms can be found in [1][24].

Chromosome representation: Each individual (binding score matrix) in the population is represented by an ensemble of *kn* real numbers representing elements in the *k* x *n* matrix, where *k* represents motif length and *n* represents number of residues. These real numbers are bound by a lower and upper limit.

Objectives: The objectives are defined to realize a motif that can best represent the characteristics of the I-A^{g7} binding motif. The dataset of each experiment in the literature gives the information whether the particular sequence is a binder or non-binder. Using this information, the numbers of true positives (TP) and true negatives (TN) determined by solutions in the population is computed. By incorporating the TPs and TNs resulting from the evaluation and by taking into the account the cumulative distance between a putative motif, *m*, representing a binding score matrix Q and the score matrix representation of the experimental motifs set, Q(**R**), we can define two objective functions f_1 and f_2 as follows:

$$\min \ f_1 = (\mathrm{FP}+\mathrm{FN}) \tag{4}$$

$$\min \ f_2 = \sum_{i=1}^{r} \left| \mathrm{Q} - \mathrm{Q}(R_i) \right| \tag{5}$$

where FP and FN denotes false positives and false negatives, respectively. And $\mathrm{Q}(R_i)$ represents the score matrix representation of an experimental motif, R_i.

Constraints: Two constraints are defined as follows:

$$c_1 = 1.0 - ((\alpha_1 * \mathrm{FP}) / \mathrm{NB}) \geq 0 \tag{6}$$

$$c_2 = 1.0 - ((\alpha_2 * \mathrm{FN}) / \mathrm{B}) \geq 0 \tag{7}$$

The α_1 and α_1 are the two control parameters preventing all binders and non-binders being recognized as binders and vice versa. NB and B correspond to the number of true non-binders and binders in the training dataset.

3 Experiments and Results

3.1 Datasets

Ten datasets [9-18] consisting of short peptides ranging from 9-30 amino acids per sequence were extracted. The datasets [9-14] together with the unpublished dataset [17] were combined and used as the training dataset after the removal of duplicates. The training set consists of 351 binding peptides and 140 non-binding peptides. Two remaining datasets, Suri [15] and Stratmann [16] were used as the testing dataset. The Stratmann test set contains only 118 binders and three non binders while Suri dataset contains 20 binders and two non binders. Due to the small number of experimentally determined non-binders, we extended the number of non-binders in this set to 1000 by generating random peptides. The generation of random peptides involved adding correct proportions of amino acids to each peptide so that the generated peptide mimics real protein peptides [20]. Of 1000 random peptides generated, at most five percent are presumed to be binders and this error estimate was taken into consideration for the performance analysis. This percentage was estimated based on the analysis of I-A^{g7} binding data given in [20].

3.2 Score Matrix Representation of the Experimental Motifs

The following scoring scheme is used to represent each experimental motif in [8-12] as a quantitative matrix. Two experimental motifs [6,7] were excluded as they do not describe many of the k-mer positions, or have assigned fewer residues for the described positions. For all the other motifs, all non tolerated positions were given a score of 0. Well tolerated residues were assigned a maximum score of 127. The preferred residues at the primary anchor positions were assigned half the maximum score. And the positions that do not carry any predefined residues were assigned one third of the maximum score. The distance to each of these score matrices is calculated and the sum of all the distances is used to optimize the objective function, f_2.

3.3 Multi-Objective Evolutionary Algorithm: NSGA-II Parameters

For the MOEA optimization runs, we used a population of 500, where each individual representing 180 real numbers bound by the limits 0 and 127. A crossover probability for the real variables is set to 1.0, and the mutation probability of 0.006, distribution index for crossover and mutation are set to 15 and 30, respectively. After 300 generations, the evolutionary process was terminated. The convergence of the algorithm to the final population is illustrated in Figure 1. Having a number of Pareto solutions allows the user to choose a best solution. Of the motif solution set, we chose the motif that gives the highest A_{ROC} for the test dataset.

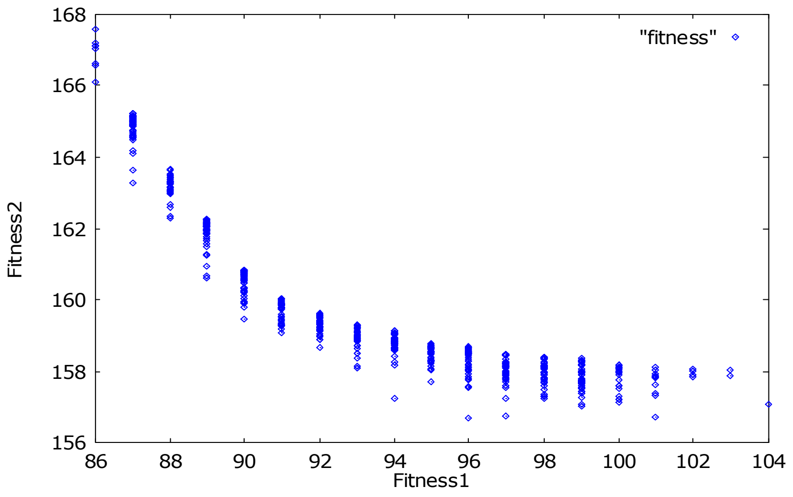


Fig. 1. Final population after 300 iterations, with the mutation probability=0.006 and cross-over probability=1.0. Fitness1 depicts the sum of FP and FN with respect to the best individuals in the population whereas Fitness2 indicates the cumulative sum of the distances to the experimental motifs and the best individuals (score matrices) in the population.

3.4 Performance Comparison

Computationally derived motifs: MEME, Gibbs Motif Sampler and Rankpep. All binders in the training set are used to derive a motif by the computational methods MEME, Rankpep, and Gibbs motif sampler. Three motifs were derived from the MEME, and the motif that performed the best was used for our experiments M-K-R-H-G-L-D-N-Y. With the Gibbs motif sampler, the residues at each position that conveyed the highest mutual information content were retained and the motif was composed N(MP)-K(V)-A(RI)-T(H)-G(A)-E(FL)-D(Q)-N(YL)-K(YV). For Rankpep, the consensus motif was W-Y-A-H-A-F-K-Y-V. Using these motifs and scoring matrices we measured the predictive performance on the Stratmann and Suri datasets combined with randomly generated non-binders. The performance (Figure 2a) was measured by the Area under the Receiver Operating Characteristics (A_{ROC}) curves (see [26]).

Experimentally derived motifs. The performance of the experimentally derived motifs was estimated by assigning a scoring scheme: anchor residues have weight 4, tolerated 2, and non-tolerated -4. Primary anchor positions were assigned weight 4 and secondary anchor positions have weight 2. The primary and secondary anchor positions were defined according to the motif descriptions by the authors. The A_{ROC} plots for the experimental motifs and the MOEA derived motif are shown in Figure 2(b). Of seven experimentally derived motifs the A_{ROC} curves of only five motifs are shown in Figure 2(b) for clarity. The A_{ROC} calculations for all motifs are given in the Table 2. Overall, the motif derived from MOEA performed better on the test dataset-than the other motifs described in literature and the computationally derived motifs by MEME, Gibbs Motif Sampler and Rankpep (Figure 2).

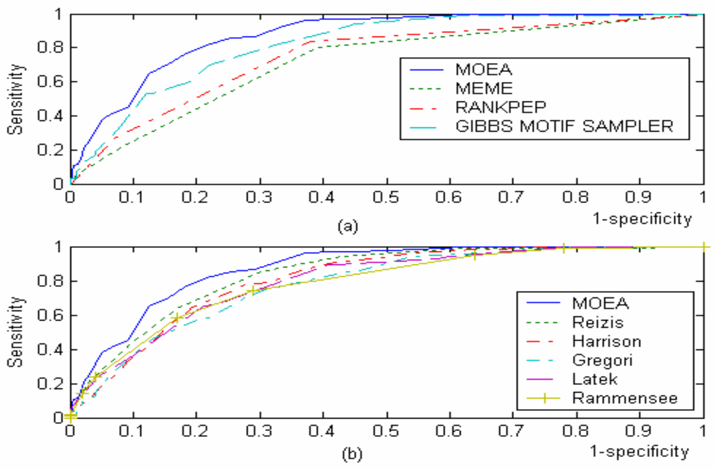


Fig. 2. a) A_{ROC} plots between theoretically derived motifs (a). b) Performance comparison between five experimentally derived motifs and MOEA-motif.

Table 2. The A_{ROC} values from predictions using each motif on the independent test dataset. $A_{ROC}>0.9$ correspond to excellent, $0.8<A_{ROC}<0.9$ to good, $0.7<A_{ROC}<0.8$ to marginal prediction accuracy. $A_{ROC}=0.5$ corresponds to random guessing, and $0.5<A_{ROC}<0.7$ to poor predictions.

Motif	**A_{ROC}**
Reizis	**0.81**
Harrison	**0.81**
Gregori	0.78
Latek	0.79
Rammensee	0.78
Reich	0.74
Amor	0.78
MEME	0.72
Gibbs	**0.81**
Rankpep	0.75
MOEA	**0.87**

Table 3. The A_{ROC} values estimated on the individual datasets used in deriving the experimental motifs by the best performing motifs in Table 2

	Motif			
Dataset	**Reizis**	**Harrison**	**Gibbs**	**MOEA**
Reizis	**0.95**	0.75	0.33	0.77
Harrison	0.68	**0.88**	0.79	0.76
Gregori	0.74	0.69	0.77	0.84
Latek	0.95	0.64	0.81	0.89
Corper	0.50	0.53	0.39	0.70
MHCPEP	0.59	0.72	0.64	0.85
Yu	0.48	0.33	0.58	0.61

The performance of the Multi-objective motif on the individual datasets used in the derivation of the experimental motifs show similar or better results (greater than 0.7 A_{ROC} value) compared to the performance of other motifs across datasets (Table 3). The performance of the motifs on their respective datasets indicated their bias towards their own datasets. Overall MOEA-derived matrix performed well across all datasets except in the case of Yu dataset, for which it gave the $A_{ROC}=0.61$. Other experimental motifs on the same dataset performed poorly with $A_{ROC}<0.61$ except for the Latek motif ($A_{ROC}>0.7$). The MOEA-derived matrix reconciles significant variances in the experimental motifs and minimizes the number of false predictions in the test dataset as compared to the performance of previously determined experimental motif.

4 Conclusions

We have proposed MOEA for deriving a consensus motif from a number of experimentally derived motifs. One of the objectives of our approach is to optimize the number of true predictions across all I-A^{g7} datasets, which was not the case with the experimentally derived motifs. The experimental motifs formulated from each independent study show biases to their own datasets - they perform well on the respective dataset, but poorly on the other datasets. The other objective is to capture the significance of each motif and combine them together so that the resulting motif can act as a consensus motif characterizing all the experimental motifs. As we can see from the results, the derived motif performed comparatively well on all the datasets. Furthermore, the MOEA evolved solution outperformed all the other computationally derived motifs demonstrating its suitability for discovery of highly accurate motifs.

Acknowledgements

This work was funded in part (MR and VB) by the NIAID, NIH, Dept. of Health and Human Services, Grant 5 U19 AI56541 and Contract HHSN266200400085C.

References

1. Deb, K. *et al.*: A Fast and Elitist Multiobjective Genetic Algorithm. IEEE Trans. on Evolutionary Computation 6 (2002) 182-197
2. Zitzler, E. *et al.*: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength of Pareto Approach. IEEE Trans. on Evolutionary Computation 3 (1999) 257-271
3. Knowles, J. D. *et al.*: Approximating the Nondominant front using the Pareto Archived evolution stratergy. Evolutionary Computation 8 (2000) 49-172
4. Fonseca, C. M. *et al.*: Genetic Algorithms for Multiobjective Optimization: Formulation, discussion and generalization. In Proc. of the fifth Intl. conference on Genetic Algorithms, S. Forrest, Ed. San Mateo, CA: Morgan Kauffman (1993) 416-423
5. Lee, S. *et al.*: Comparison of Multi-Objective Genetic Algorithms in Optimizing Q-Law-Thrust Orbit Transfers. GECCO (2005)
6. Amor,S. *et al.*: Encephalitogenic epitopes of myelin basic protein, proteolipid proteing, and myelin oligodendrocyte glycoprotein for experimental allergic en-cephalomyelitis induction in Biozzi AB/H(H-2A^{g7}) mice share an amino acid motif. J. Immunology 156 (1996) 3000-3008
7. Reich,E.P. *et al.*: Self peptides isolated from MHC glycoproteins of non-obese diabetic mice. J. Immunology 152 (1994) 2279-2288
8. Rammensee, H. *et al.*: SYFPEITHI:database for MHC ligands and peptide motifs. Immunogenetics 50 (1999) 213-219
9. Reizis,B., *et al.*: Molecular characterization of the diabetes mouse MHC class-II protein, I-A^{g7}. Int. Immunology 9 (1997) 43-51
10. Harrison,L.C. *et al.*: A peptide binding motif for I- A^{g7}, the class II major jistocompatibility complex (MHC) molecule of NOD and Biozzi AB/H mice. J. Exp. Med. 185 (1997) 1013-1021
11. Latek,R.R. *et al.*: Structural basis of peptide binding and presentation by the type I diabetes-associated MHC class II molecule of NOD mice. Immunity 12 (2000) 699-710
12. Gregori,S. *et al.*: The motif for peptide binding to the insulin-dependent diabetes mellitus-associated class II MHC molecule I-A^{g7} validated by phage display library. Int. Immunology 12(4) (2000) 493-503

13. Corper,A.L. *et al.*: A structural framework for deciphering the link between I-A^{g7} and autoimmune diabetes. Science 288 (2000) 505-511
14. Yu,B. *et al.*: Binding of conserved islet peptides by human and murine MHC class II molecules associated with susceptibility to type I diabetes. J. Immunology 30(9) 2497-506
15. Suri, A. *et al.*: In APCs, the Autologous Peptides Selected by the Diabetogenic I-A^{g7} Molecule Are Unique and Determined by the Amino Acid Changes in the P9 Pocket. J Immunol 168(3) (2002) 1235-43
16. Stratman,T. *et al.*: The I-A^{g7} MHC class II molecule linked to murine diabetes in a promiscuous peptide binder. J. Immunology 165 (2000) 3214-3225
17. Brusic,V. An unpublished dataset
18. Brusic,V., Rudy,G., Harrison,L.C. MHCPEP, a database of MHC-binding peptides: update 1997. Nucleic Acids Res. 26 (1998) 368-371
19. Bailey,T.L., Elkan,C. The value of prior knowledge in discovering motifs with MEME. Proc Int Conf Intell Syst Mol Biol. 3 (1995) 21-29
20. Pe'er I. *et al.* Proteomic Signatures:Amino Acid and Oligopeptide Compositions Differentiate Among Phyla. Proteins 54 (2004) 20-40
21. http://meme.scdc.edu/meme/website/meme.html
22. Neuwald, A. F. *et al.*: Gibbs motif sampling: detection of bacterial outer membrane protein repeats. Protein Science 4 (1995) 1618-32
23. Reche, P, A. *et al.*: Enhancement to the RANKPEP resource for the prediction of peptide binding to MHC molecules using profiles. Immunogenetics 56 (2004) 405-419
24. Coello, C. A., An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends. Congress on Evolutionary Computation, Washington: IEEE Service Center (1999) 3-13
25. Carraso-Marin, E., Kanagawa, O., Unanue, E. R., The lack of consensus for I-A^{g7}-peptide binding motifs: Is there a requirement for anchor amino acid side chain. Proc. Natl. Acad. Sci. 96 (1999) 8621-8626
26. Rajapakse, M., *et al.*: Deriving Matrix of Peptide Interactions in Diabetic Mouse by Genetic Algorithm. Proc. of 6th International Conference on Intelligent Data Engineering and Automated Learning, LNCS, Springer, 3578 (2005) 440-447

Mining Structural Databases: An Evolutionary Multi-Objetive Conceptual Clustering Methodology

R. Romero-Zaliz[1], C. Rubio-Escudero[1], O. Cordón[1], O. Harari[1], C. del Val[1], and I. Zwir[1,2]

[1] Dept. Computer Science and Artificial Intelligence, University of Granada, E-18071, Spain
{rocio, crubio, igor, ocordon, delval}@decsai.ugr.es
[2] Howard Hughes Medical Institute, Department of Molecular Microbiology, Washington University School of Medicine, St. Louis, MO 63110-1093, USA
zwir@borcim.wustl.edu

Abstract. The increased availability of biological databases containing representations of complex objects permits access to vast amounts of data. In spite of the recent renewed interest in knowledge-discovery techniques (or data mining), there is a dearth of data analysis methods intended to facilitate understanding of the represented objects and related systems by their most representative features and those relationship derived from these features (i.e., structural data). In this paper we propose a conceptual clustering methodology termed *EMO-CC* for *Evolutionary Multi-Objective Conceptual Clustering* that uses multi-objective and multi-modal optimization techniques based on Evolutionary Algorithms that uncover representative substructures from structural databases. Besides, EMO-CC provides annotations of the uncovered substructures, and based on them, applies an unsupervised classification approach to retrieve new members of previously discovered substructures. We apply EMO-CC to the Gene Ontology database to recover interesting substructures that describes problems from different points of view and use them to explain inmuno-inflammatory responses measured in terms of gene expression profiles derived from the analysis of longitudinal blood expression profiles of human volunteers treated with intravenous endotoxin compared to placebo.

1 Introduction

The increased availability of biological databases containing representations of complex objects such as microarray time series, regulatory networks or metabolic pathways permits access to vast amounts of data where these objects may be found, observed, or developed [1, 2, 3]. In spite of the recent renewed interest in knowledge-discovery techniques (or data mining), there is a dearth of data analysis methods intended to facilitate understanding of the represented objects and related systems by their most representative features and those relationship derived from these features (i.e., structural data).

Structural data can be viewed as a graph containing nodes representing objects, which have features linked to other nodes by edges corresponding to their relationships. Interesting objects in structural data are represented as substructures, which consists of subgraph partitions of the datasets [4]. Conceptual clustering techniques have been successfully applied to structural data to uncover objects or concepts that relates objects, by searching through a predefined space of potential hypothesis (i.e., subgraphs that represent associations of features) for the hypothesis that best fits the training examples [5]. However, the formulation of the search problem in a graph-based structure would result in the generation of many substructures with small extent as it is easier to explain or model match smaller data subsets than those that constitute a significant portion of the dataset. For this reason, any successful methodology should also consider additional criteria to extract better defined concepts based on the size of the substructure being explained, the number of retrieved substructures, and their diversity [4, 6]. The former are conflicting criteria that can be approached as an optimization problem. Multi-objective optimization techniques can evaluate concepts or substructures based on the conflicting criteria, and thus, to retrieve meaningful substructures from structural databases.

In this paper we propose a conceptual clustering methodology termed *EMO-CC* for *Evolutionary Multi-Objective Conceptual Clustering* that uses multi-objective and multi-modal optimization techniques. The EMO-CC methodology uses an efficient search process based on Evolutionary Algorithms [7, 8, 9], which inspects large data spaces that otherwise would be intractable. Besides, EMO-CC provides annotations of the uncovered substructures, and based on them, applies an unsupervised classification approach to retrieve new members of previously discovered substructures. We apply EMO-CC to the Gene Ontology database (i.e., the GO Project [3]) to recover interesting substructures containing genes sharing a common set of terms, which are defined at different levels of specificity and correspond to different ontologies, producing novel annotations based on them. Particularly, we use these substructures to explain inmuno-inflammatory responses measured in terms of gene expression profiles derived from the analysis of longitudinal blood expression profiles of human volunteers treated with intravenous endotoxin compared to placebo [10].

This work is organized as follows. Section 2 reviews the conceptual clustering problem. Section 3 describes the EMO-CC methodology. Section 4 shows the customization and results of applying EMO-CC to the GO database to explain gene expression profiles from the inflammatory problem. Section 5 introduces the discussion.

2 Conceptual Clustering

Cluster analysis –or simply clustering– is a data mining technique often used to identify various groupings or taxonomies in real-world databases [11]. Most existing methods for clustering are designed for linear feature-value data. However, sometimes we need to represent structural data that do not only contains descriptions of individual observations in databases, but also relationships among these observations. Therefore, mining into structural databases entails addressing both the uncertainty of which observations should be placed together, and also which distinct relationships among features best characterize different sets of observations, having in mind that, a priori, we do not know which feature is meaningful for a given relationship.

Conceptual clustering, in contrast to most typical clustering techniques [12], have been successfully applied to structural databases to uncover concepts that are embedded in subsets of structural data or substructures [4]. While most machine learning techniques applied directly or indirectly to structural databases exhibit methodological differences, they do share the same framework even though they employ distinct metrics, heuristics or probability interpretations [13, 4]: (1) *Database representation.* Structural data can be viewed as a graph containing nodes representing objects, which have features linked to other nodes by edges corresponding to their relations. A substructure consists of a subgraph of structural data [4]; (2) *Structure Learning.* This process consists of searching through the space for potential substructures, and either returning the best one found or an optimal sample of them; (3) *Cluster evaluation.* The substructure quality is measured by optimizing several criteria, including specificity, where harboring more features always increases the inferential power; sensitivity, where a large coverage of the dataset produces good generality; and diversity, where minimally overlapping between clusters generates more distinct clusters and descriptions from different angles; (4) *Database compression.* The database compression provides simpler representations of the objects in a database; and (5) *Inference.* New observations can be predicted from previously learned substructures by using classifiers that optimize their matching based on distance [14] or probabilistic metrics [5]).

3 An Evolutionary Multi-Objective Conceptual Clustering Methodology (EMO-CC)

We explicitly propose a method for each of the conceptual clustering steps mentioned:

(1) **Database representation** by using structures as graphs, where nodes correspond to database features and edges to the relationships among these features.
(2) **Structure learning** by searching in the feature space to obtain optimal substructures using an efficient multi-objective evolutionary algorithm, as well as appropriate objective definitions to guide the search relying on the NSGA-II algorithm [15]. Basic configuration of this algorithm is explained below:
Chromosome representation. EMO-CC encodes feasible substructures in the chromosomes of the algorithm population. Each chromosome is implemented

as a tree, where this representation in GAs is known as Genetic Programming (GP) [16]. This chromosome representation encodes each node and edge of the tree with a label, describing the type of feature, and an associated tag that indicates the value of such feature. The initial population consists of a set of chromosomes, each one built by choosing a random observation from the input database and extracting a subtree from its tree representation. The set of all non-dominated chromosomes of the final population represents a clustering of the given data.

Genetic operators. EMO-CC applies crossover and mutation operators with a given probability over the chromosomes composing the population of the GP. The crossover operator is performed by swapping two random subtrees, which is a classical choice in GP. The mutation operators used in our GP implementation are also classical and straightforward: (1) *Delete a leaf,* where a random leaf of the tree is selected and deleted along with the edge that connects it to the tree; (2) *Change a node,* where a random node is selected and replaced by another node belonging to the set of nodes constrained to have the same tag; and, (3) *Add a leaf,* where a random leaf is created and connected to the tree by a new edge.

Selection. EMO-CC uses a classical binary tournament selection method [17], which chooses two parent chromosomes and selects the one with the higher fitness value.

Multi-objective optimization. We consider that good substructures are those ones that maximize the *specificity* and *sensitivity* objectives. On the one hand, the specificity of a substructure is associated with its size (i.e., the number of objects and features that compose the substructure), which corresponds to the size of the tree represented in the chromosome. On the other hand, the sensitivity of a substructure is calculated as the number of instances that occur in the substructure, where an instance occur in a substructure if its tree representation is a subtree of the substructure tree. These are opposing objectives since the more specific the substructure, the less sensitive it becomes to detect new instances.

Non-dominance relationship. We select substructures that satisfy a trade-off between their specificity and sensitivity by selecting a set of solutions that are non-dominated, in the sense that there is no other solution that is superior to them in all objectives (i.e., Pareto optimal front [8, 6]). Another objective that is indirectly considered is the substructure diversity, which consists of maintaining a distributed set of solutions in the Pareto front. Therefore, to address all of these objectives our approach applies the non-dominance relationship locally, that is, it identifies all non-dominated optimal substructures that have no better solution in a neighborhood [8, 6]. We consider that two substructures are in the same neighborhood if they have at least a 50% of instances occurring in both of them calculated based on the *Jaccard's coefficient* [18].

(3) **Clustering evaluation** applying the non-dominance relationship between conflicting criteria in a neighborhood to achieve cohesive, well supported, and diverse substructures.

(4) **Compression of substructures** based on an circumstantial query, thus allowing flexible and adaptive substructures to different contexts.
(5) **Inference** by using an unsupervised fuzzy k-nearest prototype classifier that characterizes new instances based on available knowledge. It calculates the membership of a query observation x_q in a set of I previously identified substructures.

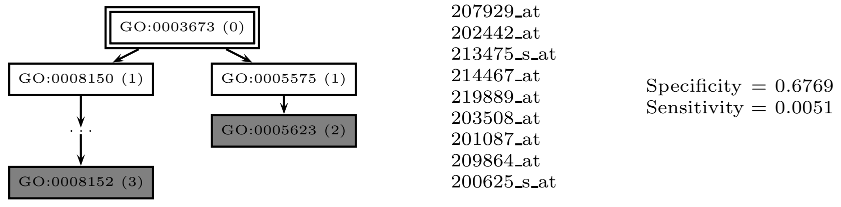
4 Application of the EMO-CC Methodology to the Gene Ontology Structural Database

Massive microarray experiments provide a wide view of the gene regulation problem; however, most of the biological knowledge extracted from these experiments include few relevant genes, some of which are difficult to be identified because of their low expression levels. Moreover, it is also difficult to distinguish among expressed genes that behave differentially between treatments, time, patients and other factors that are always hidden in typical microarray protocols (e.g., gender or age). Here we focus on the challenge of explaining these profiles and re-discover them based on independent biological information.

We therefore apply EMO-CC to discover interesting substructures in the Gene Ontology database that can explain classes composed of microarray gene profiles having similar behaviors of their expression over time, treatment, and patient. The Gene Ontology (GO) network stores one of the most powerful characterization of genes, containing three structured vocabularies (i.e., ontologies) that describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner [3]. The GO terms are organized as hierarchical networks, where each level corresponds to a different specificity definition of such terms (i.e., higher level terms are more general than lower level terms). Particularly, from the computational point of view, these networks are organized as structures called directed acyclic graphs (DAGs), which are one way routed graphs that can be represented as trees. Therefore, identifying which distinct relationships among features best characterize different sets of observations does not only have to consider the process of grouping distinct type of features, but also defining at which level of specificity they have to be represented.

4.1 EMO-CC Customization for the GO Domain

We used the GO database and compatibilized the terms with descriptions provided by Affymetrix, where each observation of the database has the following features: (1) *Name:* Affymetrix identifier for each gene in HG-U133A v2.0 set of arrays; (2) *Biological process:* List of the biological processes where a gene product is involved (e.g., mitosis or purine metabolism); (3) *Molecular function:* List of the biological functions of the gene product (e.g., carbohydrate binding and ATPase activity), which is indexed by a list of integer GO codes; and (3) *Cellular component:* List of the cellular components indicating location of gene



207929_at
202442_at
213475_s_at
214467_at
219889_at
203508_at
201087_at
209864_at
200625_s_at

Specificity = 0.6769
Sensitivity = 0.0051

(a) Chromosome (b) Genes (c) Objectives

Fig. 1. An example of a chromosome representing a cluster. (a) The tree representation, gray boxes represent the most specific GO terms of the concept of the cluster, the level of each term is shown between parenthesis. (b) The list of genes that correspond to the cluster. (c) The values corresponding to the sensitivity and specificity objective functions.

products (e.g., nucleus, telomere, and origin recognition complex), which are indexed by a list of integer GO codes.

An instance for the GO domain is redefined as the particular subset of values that constitutes a prefix tree[1] of a database observation in contrast to a subtree as in the general case. Then, an instance occurs in a substructure if a subgraph of the prefix tree that represent that instance matches with the substructure tree, where this tree contains tagged nodes with the type of feature (e.g., biological process), and the corresponding values (e.g., GO:0007165), and the edges represent relationship between features (i.e., tagged nodes).

Good substructures are those ones that result in a trade-off between sensitivity and specificity. Although, the sensitivity can be calculated based on the number of instances in a substructure, the specificity of the substructure is not linearly dependent to its size, as it was previously defined based on the number of nodes and edges because of the level component included in the GO domain. Thus, we redefine the specificity as the distance among all most specific nodes of an instance i and the closest leaf-node in the substructure S:

$$Specificity(S) = \frac{\sum_i^K \sum_u^U \frac{dist(node_u, node_i)}{level(node_i)}}{K} \tag{1}$$

where the distance is calculated as the number of edges between two nodes, the level of a node is calculated as the length of the shortest path to the root node, U is the number of leaf-nodes in substructure S, and K is the number of instances occurring in substructure S. An example of a chormosome representing a cluster concept is shown in Figure 1.

4.2 Experiments and Analysis of Results

The structural database used for the GO domain is composed of 1770 instances of genes and their GO associated terms. The population of the evolutionary

[1] Tree T' is a prefix tree of T if T can be obtained from T' by appending zero or more subtrees to some of the nodes in T'. Notice that any tree T is a prefix of itself.

algorithm is initialized by 50% of randomly chosen subtrees of the database and by another 50% of random instances. The parameters of the algorithms used for this domain are shown in Table 1. The EMO-CC approach was run ten times with different seeds and the average of these runs is reported.

Table 1. Parameters for the GO domain

Parameter	Value
Population Size	200
Number of Objective Evaluations	20000
Crossover probability	0.6
Mutation probability	0.2

4.3 Computational Analysis

We compare EMO-CC with two other methods, APRIORI and SUBDUE, all of which satisfy in some extent those features shared by machine learning methods introduced in Section 3. Although APRIORI and SUBDUE are not MO algorithms, we illustrate the obtained Pareto fronts in Figure 2 to perform fair comparisons with EMO-CC. In addition, we verify the performance of the former methods by applying some multi-objective comparison metrics, namely $\mathcal{C}$ and $\mathcal{ND}$ [19, 20]. The metric $\mathcal{C}(X', X'')$ measures the dominance relationship between the set of non-dominated solutions X' over other set of non-dominated solutions X''. The value $\mathcal{C}(X', X'') = 1$ means that all points in X'' are dominated by points in X'. The opposite, $\mathcal{C}(X', X'') = 0$, represents the situation where none of the points in X'' are covered by the set X'. The metric $\mathcal{ND}(X', X'')$ compares two sets of non-dominated solutions and gives the number of solutions of X' not equal and not dominated by any member of X''. The values obtained by the methods are shown in Table 2,

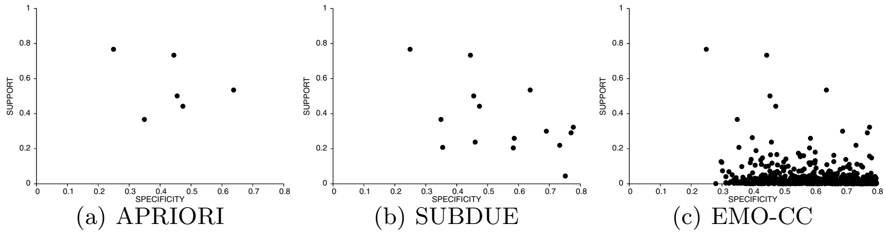


Fig. 2. Pareto fronts for the GO domain by using two conflicting objectives: specificity and sensitivity. (a) Non-dominated solutions reported by the APRIORI method. (b) Solutions recovered by the SUBDUE method. (c) Substructures recovered by the EMO-CC methodology, where more than one solution for the same specificity level indicates that they correspond to different neighborhoods.

The obtained results of applying the former metrics reveal that there is no solution obtained by EMO-CC that is dominated by APRIORI, and only one solution obtained by SUBDUE dominates solutions belonging to the Pareto front found by EMO-CC (Table 2(a)), as described by metric $\mathcal{C}$, while there is no solution of the latter method that dominates any solution from the other two approaches. Moreover, the EMO-CC method discovers more non-dominated solutions, as evaluated by metric $\mathcal{ND}$ (Table 2(b)), than both APRIORI and SUBDUE methods. The difference between the values reported by the $\mathcal{ND}$ metric from EMO- CC and those ones from APRIORI and SUBDUE (i.e., 181.89 and 171.80 vs. 1.20 and 1.60 from Table 2(b)) suggests that EMO-CC retrieves almost all solutions identified by the other methods and covers a wide set of all of optimal solutions that can be obtained in the GO domain. This is in contrast to the few solutions that are identified by the APRIORI and SUBDUE methods, but remain undetected by the EMO-CC method (i.e., 1.20 and 1.60 in average from Table 2(b)).

In addition, the EMO-CC method recovers most and more diverse solutions than those found by the APRIORI and SUBDUE methods. Particularly, our approach retrieves substructures of the Pareto optimal front containing few instances harboring several features (i.e., cohesive substructures), which were undetected by the other methods.

Table 2. Comparative evaluation of the solutions identified by APRIORI, SUBDUE and EMO-CC for the GO domain by using different metrics

(a) $\mathcal{C}$ metric

$\mathcal{C}(X', X'')$	APRIORI	SUBDUE	EMO-CC average (*stdev*)
APRIORI	-	0.00000	0.00000 (*0.00000*)
SUBDUE	0.00000	-	0.00050 (*0.00160*)
EMO-CC average (*stdev*)	0.00000 (*0.00000*)	0.08421 (*0.04438*)	-

(b) $\mathcal{ND}$ metric

$\mathcal{ND}(X', X'')$	APRIORI	SUBDUE	EMO-CC average (*stdev*)
APRIORI	-	1	1.20 (*0.42*)
SUBDUE	13	-	1.60 (*1.17*)
EMO-CC average (*stdev*)	181.80 (*11.99*)	171.80 (*11.62*)	-

Biological results analysis using gene expression profiles. We consider 24 independent classes containing gene expression profiles derived from the analysis of 48 GeneChips® HG-U133A v2.0 from Affymetrix Inc., corresponding to an inflammatory response study performed on human volunteers treated with intravenous endotoxin compared to placebo [10]. The data has been acquired from samples taken from human blood to eight patients over time at 0, 2, 4, 6, 9 and 24 hours, where four had been treated with intravenous endotoxin (i.e., patients 1 to 4) and four with placebo (i.e., patients 5 to 8). We will use these gene expression profiles for validating the substructures detected by EMO-CC, or, in other words, which are explained by these substructures.

Table 3. Clusters derived from the GO information by EMO-CC intersecting significantly with class #13 from the gene expression information. Solid lines separate groups of clusters which GO information is not related, while dashed lines separate clusters within these groups, as shown in Figure 3.

#Substr.	Biological process	Molecular function	Cellular component
179	GO:0006915 apoptosis (level: 6)		GO:0005887 integral to plasma membrane (level: 4)
536	GO:0007165 signal transduction (level: 4)		GO:0016021 integral to membrane (level: 3)
759	GO:0007165 signal transduction (level: 4)		GO:0005887 integral to plasma membrane (level: 4)
89	GO:0007154 cell communication (level: 3)		GO:0016021 integral to membrane (level: 3)
256	GO:0007154 cell communication (level: 3) GO:0050875 cellular physiological process (level: 3)		GO:0016021 integral to membrane (level: 3)
380	GO:0007165 signal transduction (level: 4) GO:0050875 cellular physiological process (level: 3)		GO:0016021 integral to membrane (level: 3)
607		GO:0004871 signal transducer activity (level: 2)	GO:0016021 integral to membrane (level: 3)

For example class #13 is described by several substructures (Table 3). Significantly, these descriptions are based on different types of descriptions (e.g., process and cellular components) that belong to different levels of the GO structure (e.g., level 6 or level 4). These diverse substructures are optimal in the sense that belong to the Pareto optimal front (Figure 2) between specific and sensitive descriptions. The effect of the substructures on the explained class #13 can be visualized in (Figure 3).

EMO-CC, as a machine learning method (see Section 3 (4)), compresses those substructures that explain an expression profile from the same point of view to provide a summarized explanation of this phenomena (Table 3). For example, substructures #89 and #216 are compressed because they are indistinguishable for the class corresponding to the expression profile #13, while substructure #179 describes it from a very different point of view and is preserved as a diverse solution. This compression is dynamic because substructures are re-grouped in a context-dependent fashion, where the context corresponds to an explained class and a different classification can produce a distinct substructure association (e.g., substructures #89 and #216 are indistinguishable for class #13, while may be not the case for other class of microarray or clinical experiments). Notably, this

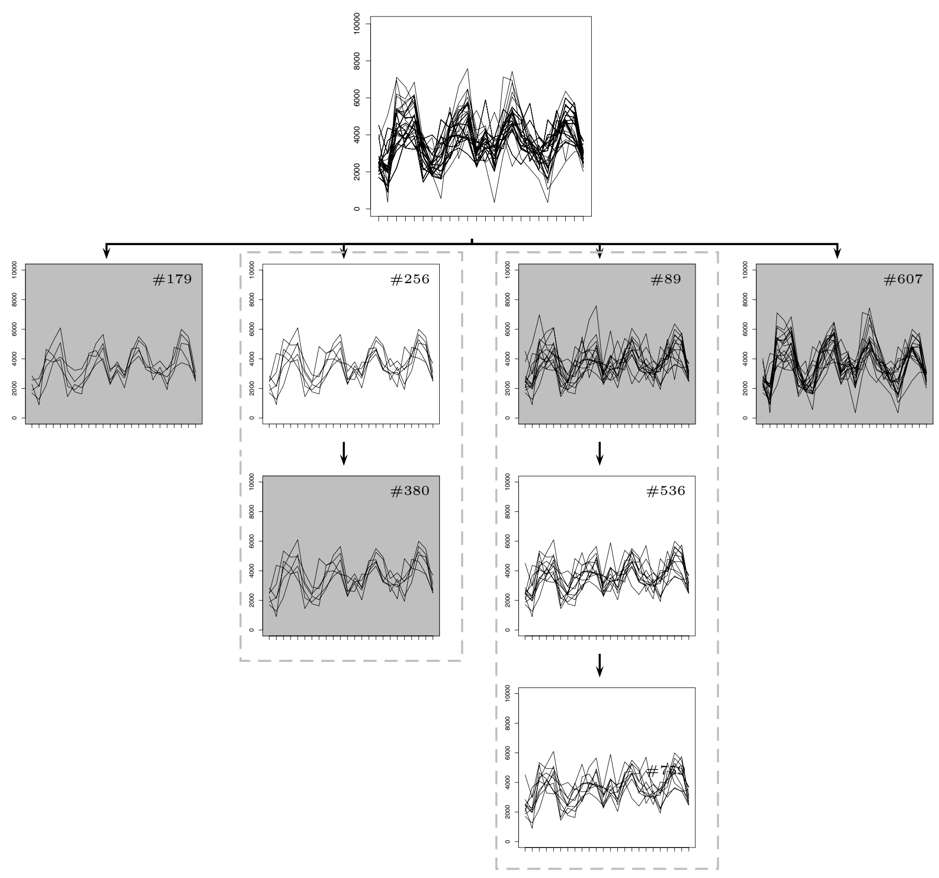


Fig. 3. The effects of the explanation of the expression class #13 based on the GO substructures identified by EMO-CC. The dashed rectangle illustrated the local application of the non-dominance relationship within a class, and the summarization of two indistinguishable substructures for this class. Grey filled graphs correspond to the compressed substructures of Table 3.

classification is performed based on completely external information provided by GO database, instead of the levels of expression.

In addition, EMO-CC applies an unsupervised inferential approach (see Section 3 (5)) which calculates the membership of a query observation x_q in a set of I previously identified substructures, to classify new instances. Since the obtained substructures are not disjoint, a given observation may belong to more than one cluster.

The unsupervised inferential mechanism of EMO-CC allows to identify new genes belonging to a particular expression profile. This is exemplified by the gene `212659_s_at`, which was recovered by its proximity to substructure #824 and shows a similar expression pattern to the genes of class #17 (Figure 4), but was ignored by the statistical methods used to recover differentially expressed genes

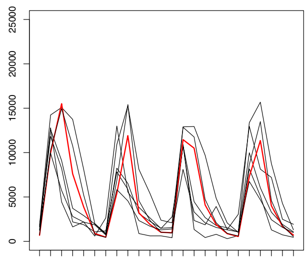


Fig. 4. Expression of Substructure B #824 where gene product `212659_s_at` is classified. The observation classified is highlighted.

[10]. It is noteworthy that this gene was not identified by its similarity with the centroid of the expression class #17, but from an independent substructure provided by EMO-CC.

5 Discussion

Unlike typical clustering techniques, conceptual clustering methods have been successfully applied to structural information in order to reveal hidden concepts by searching through a predefined space of potential hypothesis. However, the formulation of the search problem in a biological network would often result in a conflicting paradigm. On the one hand, generating a large number of substructures, each containing a very small number of genes that share all considered features, makes it hard to find commonalities among similarly regulated genes. On the other hand, generating a small number of groups in which their members share a limited number of features, would fail to discriminate between members of a molecular pathway.

In order to tackle these problems, we proposed the EMO-CC methodology that identifies conceptual clusters and classifies co-regulated genes based on multiple features that characterizes them, including functional descriptions, molecular processes and cellular components, at different levels of specificity.

EMO-CC allows gene membership to more than one substructure by using a flexible classifier [14, 21], thus, explicitly treating the substructures as hypotheses, that can be tested and refined [5]. Moreover, these hypotheses can produce novel annotations among different types of features at multiple specificity levels, which explain co-regulation phenotypes and can be used to conduct gene-wide searches.

Also, EMO-CC considers gene expression as one independent feature, thereby allowing classification of genes even in the absence of its expression. Moreover, EMO-CC minimizes the number of substructures by using a flexible compression strategy that groups similar substructures based on their ability to describe gene profiles derived from different experimental conditions (e.g., microarray expression, or Chip-on-Chip binding occupancy).

Our proposed methodology is applicable to a wide set of domains, being easily to customize to particular problem, and may be an appropriate white-box technique to uncover rear and unknown patterns in structural databases. Particularly, this guideline can be easily extended to more complex networks comprising protein-protein or different regulatory interactions [1, 2].

References

1. Siripurapu, V., Meth, J., Kobayashi, N., Hamaguchi, M.: Dbc2 significantly influences cell-cycle, apoptosis, cytoskeleton and membrane-trafficking pathways. Journal of Molecular Biology **346** (2005) 83–89
2. Nikitin, A., Egorov, S., Daraselia, N., Mazo, I.: Pathway studio–the analysis and navigation of molecular networks. Bioinformatics **19** (2003) 2155–2157
3. Consortium, T.G.O.: Gene ontology: tool for the unification of biology. Nature Genet. **25** (2000) 25–29
4. Cook, D., Holder, L., Su, S., Maglothin, R., Jonyer, I.: Structural mining of molecular biology data. IEEE Engineering in Medicine and Biology, special issue on Advances in Genomics **4** (2001) 67–74
5. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
6. Ruspini, E., Zwir, I.: Automated generation of qualitative representations of complex object by hybrid soft-computing methods. In Pal, S., Pal, A., eds.: Pattern Recognition: From Classical to Modern Approaches, Singapore, World Scientific Company (2001) 453–474
7. Back, T., Fogel, D., Michalewicz, Z., eds.: Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol, UK (1997)
8. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc. (2001)
9. Coello-Coello, C., Veldhuizen, D.V., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer (2002)
10. Romero-Zaliz, R., Cordón, O., Rubio-Escudero, C., Zwir, I., Cobb, J.: (A multi-objective evolutionary conceptual clustering methodology for gene annotation from networking databases) Submited.
11. Duda, R., Hart, P., Stork, D.: Pattern Classification (2nd Edition). Wiley-Interscience (2000)
12. Der, G., Everitt, B.: A handbook of statistical analyses using SAS. CHAPMAN-HALL (1996)
13. Cheeseman, P., Oldfors, R.W.: Selecting models from data. Springer-Vlg (1994)
14. Bezdek, J.: Fuzzy clustering. In Ruspini, E., Bonissone, P., Pedrycz, W., eds.: Handbook of Fuzzy Computation, Institute of Physics Press (1998) f6.1:1–f6.6:19
15. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
16. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
17. Goldberg, D.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley (1989)
18. Jaccard, P.: The distribution of flora in the alpine zone. The New Phytologist **11** (1912) 37–50

19. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation **3** (1999) 257–271
20. Romero-Zaliz, R., Zwir, I., Ruspini, E.: Generalized Analysis of Promoters (GAP): A method for DNA sequence description. In: Applications of Multi-Objective Evolutionary Algorithms. World Scientific (2004) 427–450
21. Gasch, A., Eisen, M.: Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. Genome Biology **3** (2002)

Optimal Selection of Microarray Analysis Methods Using a Conceptual Clustering Algorithm

C. Rubio-Escudero[1], R. Romero-Záliz[1], O. Cordón[1], O. Harari[1], C. del Val[1], and I. Zwir[1,2]

[1] Department of Computer Science and Artificial Intelligence, C/Daniel Saucedo Aranda s/n, Granada 18071, Spain
{crubio, rocio, ocordon, oharari, delval, zwir}@decsai.ugr.es
[2] Howard Hughes Medical Institute, Washington University School of Medicine, St. Louis, MO

Abstract. The rapid development of methods that select over/under expressed genes from microarray experiments have not yet matched the need for tools that identify informational profiles that differentiate between experimental conditions such as time, treatment and phenotype. Uncertainty arises when methods devoted to identify significantly expressed genes are evaluated: do all microarray analysis methods yield similar results from the same input dataset? do different microarray datasets require distinct analysis methods?. We performed a detailed evaluation of several microarray analysis methods, finding that none of these methods alone identifies all observable differential profiles, nor subsumes the results obtained by the other methods. Consequently, we propose a procedure that, given certain user-defined preferences, generates an optimal suite of statistical methods. These solutions are optimal in the sense that they constitute partial ordered subsets of all possible method-associations bounded by both, the most specific and the most sensitive available solution.

1 Introduction

Advances in molecular biology and computational techniques permit the systematical study of molecular processes that underlie biological systems [1]. Particularly, microarray technology has revolutionized modern biomedical research by its capacity to monitor changes in RNA abundance for thousands of genes simultaneously [2].

To address the statistical challenge of analyzing these large data sets, new methods have emerged ([3], [4], [5], [6], [7] and many others). However, there is a dearth of computational methods to facilitate understanding of differential gene expression profiles (e.g., profiles that change over time and/or over treatments and/or over patient) and to decide which is the most reliable method to identify differences across profiles.

We investigated the performance of several commonly used statistical methods, including T-Tests [4], Permutation Tests [5], Analysis of Variance [6] and Repeated Measures ANOVA [7], in identifying differential expression profiles that change over time, treatments and phenotype. We found that these methods do not identify all observable distinct profiles. Moreover, none of them subsumes the results obtained by the other methods.

In view of these results, we propose a conceptual clustering method [8], [9], [10], devoted to discover optimal associations of microarray analysis methods in an effort to identify differential gene expression profiles.

2 Methods

We propose a conceptual clustering approach [8], [9], [10] devoted to identify optimal associations among microarray analysis methods in an effort to identify differential expression profiles (Fig. 1). This approach consists of six phases: (1) preprocessing of the dataset; (2) identification of differentially expressed genes by application of several statistical methods; (3) arrangement of a lattice structure containing all possible associations of the statistical methods applied; (4) association of differentially expressed genes into differential profiles by clustering genes that change their expression over time, patient and/or treatment; (5) evaluation of the performance of the method-associations based on their specificity and sensitivity in the identification of previously detected differential profiles, using multiobjective optimization techniques [11], [12]. We create a set of method association rules based on the learned mappings of differential profiles into method-associations, [13]; (6) finally, we are able to predict optimal method-associations to identify differential profiles in new microarray datasets by use of the method association rules.

2.1 Identification of Differentially Expressed Genes

We perform the retrieval of differentially expressed genes from one experimental condition to the other/s by application of several statistical techniques [3], [14], harboring Student's T-Test proposed in [4], including some of the variants the method poses to distinguish changes in the abundance of RNA occurring over both treatment and time; Permutation Test described in [5], also including a time approach; Analysis of Variance described in [6]; and Longitudinal Data approach by using Repeated Measures Analysis of Variance described in [7].

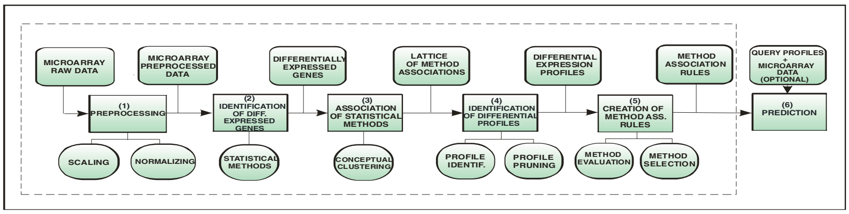

MICROARRAY RAW DATA
MICROARRAY PREPROCESSED DATA
DIFFERENTIALLY EXPRESSED GENES
LATTICE OF METHOD ASSOCIATIONS
DIFFERENTIAL EXPRESSION PROFILES
METHOD ASSOCIATION RULES
QUERY PROFILES + MICROARRAY DATA (OPTIONAL)
(1) PREPROCESSING
(2) IDENTIFICATION OF DIFF. EXPRESSED GENES
(3) ASSOCIATION OF STATISTICAL METHODS
(4) IDENTIFICATION OF DIFFERENTIAL PROFILES
(5) CREATION OF METHOD ASS. RULES
(6) PREDICTION
SCALING
NORMALIZING
STATISTICAL METHODS
CONCEPTUAL CLUSTERING
PROFILE IDENTIF.
PROFILE PRUNING
METHOD EVALUATION
METHOD SELECTION

Fig. 1. Graphical representation of the methodology. The squared boxes represent the phases of the methodology, the round cornered boxes correspond to the input/output data at each step, and the ellipses the operations performed at each phase.

2.2 Detection of Method-Associations

We arrange a lattice containing all potential associations of the statistical methods used to retrieve differentially expressed genes (Fig. 2). The methods are associated as:

$$M = \{M^1, M^2, M^n, M^1 \oplus M^2, M^1 \oplus M^3, \ldots, M^1 \oplus M^2 \oplus \ldots \oplus M^n\}, \qquad (1)$$

where $\oplus$ is a classical set operator (e.g., the union ($\bigcup$) or the intersection ($\bigcap$)) applied to the sets of genes retrieved by each method, and M^1 corresponds to T-Test, M^2 to T-Test considering time, M^3 Permutation Test, M^4 Permutation Test con sidering time, M^5 ANOVA over treatment, M^6 ANOVA over time, M^7 ANOVA over treatment and time, M^8 RMANOVA over treatment, M^9 RMANOVA over time and M^{10} RMANOVA over treatment and time.

The lattice containing all potential method-associations, *M,* is structured from top (i.e., intersection of all methods) to bottom (i.e., union of all methods) [15]. Each node in the lattice ($M^i \in M$) is applied to the microarray dataset (*D*) retrieving the set of differentially expressed genes that are recognized by the method or method-associations in such node $(M^i(D))$.

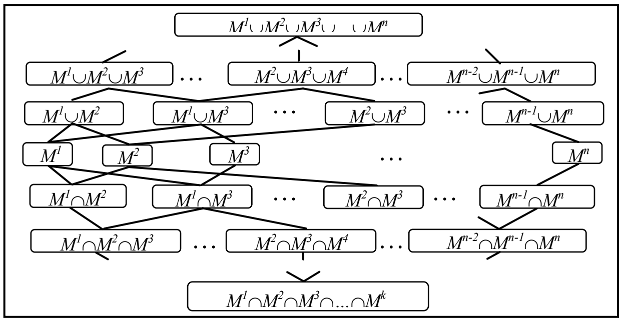


Fig. 2. Lattice structure containing all statistical methods potential associations

2.3 Identification of Differential Profiles

The set of genes previously identified in Section 2.2 serves as a means to create differential expression profiles (i.e., sets of genes with coordinate changes in RNA abundance) between treatment P_T, control P_C and subject. The applied representation (Fig. 3) allows us to identify different pattern behavior among patients inside the same experimental group, since this information may be missed if patients in the same experimental group were not plotted individually.

We clustered separately genes in treatment and control groups. Therefore, genes belonging to a cluster in treatment, P_T, can fit in more than one cluster in control, P_C, and vice versa. We apply the *K*-means clustering algorithm [16] and identify differential profiles denoted as $(P_T P_C)$, which are pairwise relationships between profiles, P_T

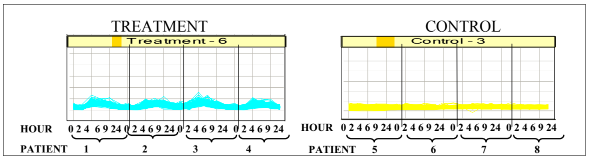


Fig. 3. The expression profiles have been represented separately for each experimental group and patients arranged individually

and P_C, from treatment and control experiments, respectively. This relationship is defined as the significant intersection of genes between P_T and P_C, which is constrained by a threshold based on the typical statistical power of 80%.

2.4 Creation of Method Association Rule

We create a set of method association rules that, given a set of differential profiles queried by the user, suggests the most appropriate method-associations capable to retrieve them. The method association rules are created based on the lattice structure from Section 2.2, containing all potential method-associations, and the set of all possible differential profiles *P* from Section 2.4 defined as $P = \{(P_T P_C)_1, ..., (P_T P_C)_l\}$ where $(P_T P_C)_j \in P$ represents each of the differential profiles present in *P*.

2.4.1 Method-Association Performance Evaluation

We evaluate the performance of the method-associations $M^i \in M$ for the query profiles $X^S = (x_1, .., x_s)$, over two objectives: *specificity* and *sensitivity*

$$Specificity = TN/(TP + FN) \qquad Sensitivity = TP/(TP + FN)\,, \tag{2}$$

where TP stands for True Positives (i.e., genes exhibiting profile $x_u \in X^S$, which have been successfully retrieved by the applied method-association M^i), TN stands for True Negatives (i.e., genes exhibiting profile $x_u \notin X^S$ and not retrieved by M^i), FP stands for False Positives (i.e., genes exhibiting profile $x_u \notin X^S$ and retrieved by M^i) and FN stands for False Negatives (i.e., genes exhibiting profile $x_u \in X^S$ and not retrieved by M^i). These four factors are calculated as:

$$TP = \frac{\varphi^u \cap \eta^i}{\varphi^u} \; TN = \frac{(D-\varphi^u) \cap (D-\eta^i)}{(D-\varphi^u)} \; FP = \frac{(D-\varphi^u) \cap \eta^i}{(D-\varphi^u)} \; FN = \frac{\varphi^u \cap (D-\eta^i)}{\varphi^u}, \tag{3}$$

where φ^u represents the genes in the microarray set *D* that exhibit the queried profile $x_u \in X^S$, and $\eta^i = M^i(D)$, the genes from *D* retrieved by the method-association M^i.

2.4.2 Method-Association Selection

We evaluate the method-associations in *M* based on their specificity and sensitivity. These two objectives are always conflicting, so we use a multiobjective optimization technique to maximize them, allowing us to detect all optimal methods-associations in M for the query profiles X^S [11], [12]. We define objectives (O_1, O_2) corresponding to specificity and sensitivity respectively.

2.4.3 Creation of a Set of Method Association Rules

We use the non-dominated method-associations described in Section 2.4.2 to create the method association rules $R = \{R^1, \ldots, R^k\}$ where $R^f \in R$ is defined as:

$$R^f : \text{IF } x_1 \text{ IS } (P_T P_C)_1^f \text{ AND}, \ldots, \text{AND } x_s \text{ IS } (P_T P_C)_s^f \text{ THEN } z^f \text{ IS } M^i \text{ WITH } C^f , \tag{4}$$

where $(x_1, \ldots, x_s)$ are the profiles X^S queried by the user; $(P_T P_C)_1^f, \ldots, (P_T P_C)_s^f \in P$; $z^f \in M$ is the appropriate method-association to retrieve X^S according to rule R^f ; and C^f denotes a measure of the specificity/sensitivity levels for z^f , defined as:

$$C^f = \frac{(w_1 * O_1(M^i)) + (w_2 * O_2(M^i))}{w_1 + w_2}, \tag{5}$$

where w_1 and w_2 are the weights associated to (O_1, O_2) respectively. These values are provided by the user based on the relevance of each of these objectives for the particular study. If no values are given, the standard (0.5, 0.5) are used.

2.5 Prediction Using Method Association Rules

The prediction phase works at two levels depending on the given input. If the input is a microarray data set *D*', our methodology will provide the differential expression profiles *P'* in the data set along with the optimal method-associations to retrieve such profiles. It might be the case that some of the differential profiles *P'* uncovered from *D*' were not included in the set of differential profiles *P* already learned by the methodology. Consequently, the information provided as input will be used to update *P* and *R*. If the input is a set of query profiles X^S , the output will consist of the optimal method-association M^h for X^S at a certain C^f value. To obtain these outputs, we apply *matching* and *inference* operations to the method association rule set [17].

Given an association rule set $R = \{R^1, \ldots, R^k\}$, for the differential profiles provided as the query set $X^S = (x_1, \ldots, x_s)$, we define the matching degree Q of $x_u \in X^S$ with the *if-part* of the association rule R^f as:

$$Q(x_u, (P_T P_C)_u^f) = 1 - \left\| \overline{x_u} - \overline{(P_T P_C)_u^f} \right\|, \tag{6}$$

with $\| \ \|$ being the Euclidean distance, and $\overline{(P_T P_C)}$ the centroids of the profiles.

Therefore, given a set of query profiles X^S , we define the strength of activation of the *if-part* of the rule R^f as:

$$R^f(X^S) = \min(Q(x_1, (P_T P_C)_1^f), \ldots, Q(x_s, (P_T P_C)_s^f)) . \tag{7}$$

Let $h^f(R^f(X^S), C^f)$ denote the *degree of association* of the query profiles X^S with the method-association M^i according to rule R^f and the specificity/sensitivity level C^f. This degree is obtained by applying a product operator between $R^f(X^S)$ and C^f. The optimal method-association for the queried profiles X^S is defined as:

$$M^i / h^i(R^i(X^S), C^i) = \max_{f \in k} h^f(R^f(X^S), C^f) . \tag{8}$$

3 Results

We apply our procedure to a data set derived from longitudinal blood expression profiles of human volunteers treated with intravenous endotoxin compared to placebo. We expect to identify molecular pathways that provide insight into the host response over time to systemic inflammatory insults, as part of a Large-scale Collaborative Research Project sponsored by the National Institute of General Medical Sciences (www.gluegrant.org) [18].

The data were acquired from blood samples collected from eight normal human volunteers, four treated with intravenous endotoxin (i.e., patients 1 to 4) and four with placebo (i.e., patients 5 to 8) [18]. Complementary RNA was generated from circulating leukocytes at 0, 2, 4, 6, 9 and 24 hours after the i.v. infusion and hybridized with GeneChips® HG-U133A v2.0 from Affymetryx Inc., containing a set of 22283 genes.

3.1 Identification of Differentially Expressed Genes

The statistical methods harbored have been applied using the standard *p-value* $\sigma = 0.05$. The number of differentially expressed genes retrieved by each of the methods from the original set of genes is M^1-10942 genes, M^2-7841, M^3-3904, M^4-8023, M^5-13151, M^6-4588, M^7-6070, M^8-8557, M^9-3995, M^{10}-3367. These values show the number of significant genes retrieved by each of the statistical methods ranges in a wide rank. Moreover, the concordance rates also vary widely, in-

Table 1. Coincidence between methods in the retrieval of genes. The number in each cell represents a ratio of coincidence between genes retrieved by the statistical method in that column and the genes retrieved by the statistical method in that row relative to the total number of genes retrieved by the method in the row ($(Row \cap Column) / Row$).

%	M^1	M^2	M^3	M^4	M^5	M^6	M^7	M^8	M^9	M^{10}
M^1	--	92.20	52.29	75.05	96.48	69.23	85.55	70.06	61.33	50.52
M^2	56.06	--	34.07	57.84	85.27	59.54	71.11	62.64	50.57	42.98
M^3	82.19	88.07	--	96.24	94.77	57.35	78.75	72.87	56.86	46.73
M^4	67.22	85.19	54.84	--	95.16	55.49	73.65	70.20	51.49	42.83
M^5	55.20	77.80	33.45	58.94	--	50.28	66.72	66.38	46.42	38.93
M^6	59.04	83.51	31.11	52.84	77.30	--	89.63	56.56	60.64	49.38
M^7	58.36	79.79	34.18	56.10	82.05	71.70	--	62.34	57.23	49.07
M^8	57.36	84.34	37.96	64.17	95.96	54.30	74.80	--	49.62	40.51
M^9	62.10	84.21	36.63	58.21	84.74	72.00	84.95	61.36	--	72.31
M^{10}	59.56	83.34	35.05	56.37	82.72	68.26	84.80	58.34	84.19	--

dicating that none of the methods subsumes the others (Table 1)(e.g., from the genes retrieved by M^3, only 31.11% are also retrieved by M^5, and 52.29% by M^1).

3.2 Association of Statistical Methods

The lattice arranged in this particular work contains all potential combinations of union and intersection of the ten statistical methods applied. Thus, M' is defined as

$$M = \{M^1, M^2, ..., M^{10}, M^1 \oplus M^2, M^1 \oplus M^3, \ldots,$$
$$M^2 \oplus M^3, \ldots, M^1 \oplus M^2 \oplus M^3, \ldots, M^1 \oplus M^2 \oplus M^3 \oplus \ldots \oplus M^9 \oplus M^{10}\}$$

We found that there is a relationship between the statistical methods and the differential profiles they are able to identify (see Section 2.2), having differential profiles identified by some methods and not by others. For example, the differential profile in (Fig. 4(a)) harbors 29 genes in our dataset D and is only retrieved by those statistical methods that take into account the time factor (e.g., M^2, which retrieves more than 90% of these genes). This happens because the statistical methods that consider the treatment vs. control factor make an average of the expression values from patients 1 and 2 with those of patients 3 and 4 by considering them as replicas. Consequently, the differential behavior between them is lost.

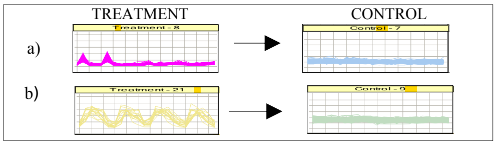


Fig. 4. Examples of differential profiles only identified by some of the statistical methods

3.3 Identification of Differential Profiles

The expression profiles have been represented separately for each experimental group (Section 2.3), and patients arranged individually. In our current problem, with eight patients, four treated with intravenous endotoxin (i.e., patients 1 to 4) and four with placebo (i.e., patients 5 to 8), and data retrieved over time at hours 0, 2, 4, 6, 9 and 24, each profile is represented by 24 consecutive time points (see Fig. 5).

The differential profiles extracted from the treatment group show different levels of expression change. For example, there are sets of genes sharing very high variations in the levels of expression (e.g., profiles 15, 19, 21, and 22 in Fig. 5). In addition, some other profiles show differential characteristics for the patients (e.g., profiles 8 and 16 in Fig. 5). In the control group, the profiles are more homogeneous than in the treatment group.

Typically, testing the coincidence among different data sources and clustering methods serves as a tool to investigate the validity of the identified groupings [19]. We follow this guideline to increase the confidence in the obtained differential profiles. Therefore, we calculate the coincidence between our retrieved differential profi-

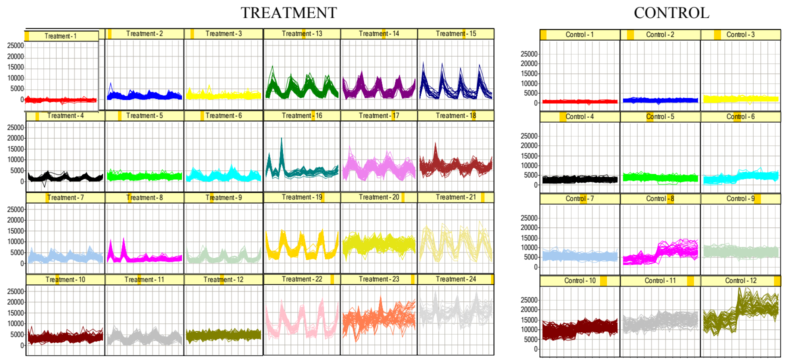


Fig. 5. Representation of the differential profiles obtained separately for the treatment and control groups using the statistical methods applied in the current work

les and external information provided by the Gene Ontology database [20]. To address this problem we developed an evolutionary multiobjective conceptual clustering methodology (R.R.Z., C.R.E., O.C., J.P.C., and I.Z., manuscript in preparation) that extracts clusters composed of features such as biological processes, molecular functions and cellular components defined at different specificity levels, and compare these clusters with our differential profiles by using a coincidence index test based on the hypergeometric distribution [9], [10], [19].

3.4 Creation of Method Association Rules

We have arbitrarily selected six profiles (i.e., $(P_T P_C)_1, \ldots, (P_T P_C)_6$) identifying a total of 1395 genes in our dataset D and plotted as treatment clusters 2, 3, 4, 5, 10 and 12 in Fig. 5. These profiles represent genes exhibiting non-uniform behavior for distinct patients in the treatment group, and genes with changes in a level of expression smaller than 5000. We applied our methodology to find the optimal method-associations M^i to retrieve them.

3.4.1 Method Association Performance Evaluation

The results of the evaluation of the method-associations contained in the lattice M' for the differential profiles are shown in Table 2, where the information relative to the sensitivity and specificity levels for the application of the most representative method-associations over D is also specified. On the one hand, we observe that the union set of the genes obtained by seven of the statistical methods evaluated (i.e., methods $M^2, M^3, M^5, M^6, M^7, M^8, M^{10}$) contains the 1395 genes desired (i.e., sensitivity value of 1) but with a low level of specificity (i.e., value of 0.369). On the other hand, the intersection set of genes obtained by the same seven statistical methods has a very low level of sensitivity (i.e., only 95 out of the 1395 genes were retrieved), whereas the value for specificity is very high. In between these two extremes we see some other method-associations which evaluation reveal trade-off solutions between the specificity and sensitivity objectives (Table 2).

3.4.2 Method Association Selection

Once the method-associations *M* have been evaluated, we search for the non-dominance relations in their applications to the microarray dataset *D*. The decision is based on the levels of specificity and sensitivity in Table 2. The Pareto optimal front conformed by this set of non-dominated method-associations is represented in Fig. 6.

Table 2. Specificity and sensitivity values for the method-associations. The non-dominated solutions are pointed out with a star.

Methods	Specificity	Sensitivity
M^2	0.611	0.707
M^3	0.826	0.205
M^5	0.448	0.785
* M^6	0.813	0.447
* M^7	0.747	0.587
M^8	0.625	0.537
* M^{10}	0.859	0.322
$M^2 \cap M^3$	0.803	0.432
* $M^2 \cup M^3$	0.618	0.866
* Union of ($M^2, M^3, M^5, M^6, M^7, M^8, M^{10}$)	0.3690	1
* Intersection of ($M^2, M^3, M^5, M^6, M^7, M^8, M^{10}$)	0.983	0.066

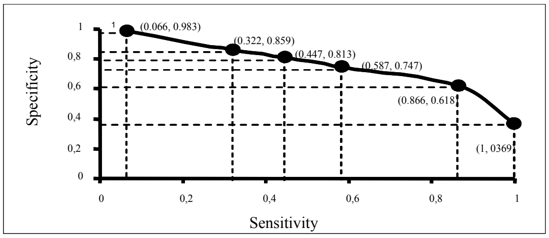


Fig. 6. Results of the evaluation of the method-associations contained in the lattice *M*' for the six selected differential profiles

3.4.3 Creation of Method Association Rules

The set of method association rules is created based on the evaluated profiles (i.e., $(P_T P_C)_1, \ldots, (P_T P_C)_6$), and the method-associations M^i present in the Pareto optimal front of non-dominated solutions. The weights (w_1, w_2) associated to the objectives (O_1, O_2) are set to (0.5, 0.5) to calculate the specificity/sensitivity measure C^f. We illustrate two association rules extracted from the evaluation of M' over the former profiles, which have the following form:

$$R^1 : \text{IF } x_1 \text{ IS } (P_T P_C)_1^1 \text{ AND }, \ldots, \text{ AND } x_6 \text{ IS } (P_T P_C)_6^1 \text{ THEN } Z^1 \text{ IS } M^6 \text{ WITH } C^1$$

where C^f is calculated based on the specificity/sensitivity levels obtained on the application of such method over $(P_T P_C)_1$,..., $(P_T P_C)_6$ profiles (Table 2):

$$C^1 = (0.5 * 0.813) + (0.5 * 0.447) / (0.5 + 0.5) = 0.631$$

and: R^2 : IF x_1 IS $(P_T P_C)_1^2$ AND,…,AND x_6 IS $(P_T P_C)_6^2$ THEN Z^2 IS $M^2 \cup M^3$ WITH C^2

where C^2 is defined as: $C^2 = (0.5 * 0.618) + (0.5 * 0.866) / (0.5 + 0.5) = 0.742$

3.5 Prediction Using Method Association Rules

To evaluate the ability of our computational approach to retrieve differential profiles, we have randomly selected 100 query sets X^S containing a random number of differential profiles from the 24 actually available. Using the method association rules created, and averaging the results, we obtained an 86.92% of overall performance measurement [21] as a particular correlation coefficient implementation.5 Prediction using method association rules

4 Discussion

The emergence of microarray technology as a standard tool for biomedical research has necessarily led to the rapid development of specific analytical methods to handle these large data sets. Despite the multiplicity of methods devoted to identify differentially expressed genes, there is a dearth of computational methods intended to optimize use of a particular method or suite of methods. Our motivation was to address two frequently asked questions: 1) do all methods retrieve the same results with the same set of input data, and 2) are the results from methods which retrieve a smaller amount of genes subsumed in the results of methods retrieving a larger amount of genes? We have shown herein how commonly used statistical methods yield different results for the same data input: each statistical method applied neither identifies all observable differential profiles, nor subsumes the results obtained by the other methods (see Tables 1 and 2). Our method also addresses another common conundrum, specifically the need for computational methods to facilitate understanding of differential gene expression profiles, to establish comparisons among them, and to decide which the most reliable method to identify informational profiles is. In this context we propose a procedure that generates optimal associations of microarray analysis methods for the set of data being analyzed, based on the differential expression profiles exhibited by the genes in the dataset.

The generation of the optimal method-associations is based on a set of previously obtained method association rules between differential profiles and the optimal method-associations to identify them. The methodology proposed is valid for either providing the optimal method-associations for a set of query profiles, or identifying all differential profiles in a given set of microarray data, suggesting the optimal method-associations for them and updating the set of possible profiles used for prediction. Although we have applied our procedure to a time-course structured experiment, we have to take into account that time-course experiments constitute more general cases than simpler microarray problems where time is not a factor and microarray samples are taken as single data points. Therefore, the methodology presented is also useful for simpler microarray experiments with single data points.

This approach presents various advantages over the standard analytical methods usually applied to microarray experiments. First, it permits combining the results of independent analytical methods for microarray experiments. Our proposal consists of a conceptual clustering technique that combines the advantages of the methods applied. The combination of the union and intersection operators also provides the possibility of querying negative samples (i.e., genes which exhibit a given profiles but not others). Second, it permits interaction with the user in the selection of differentially expressed profiles, where the user provides the differential profiles queried from the set of microarray data and receives the optimal combination of statistical methods to retrieve the genes exhibiting those profiles. Third, the representation used for the profiles is optimal, as plotting the patients sequentially presents advantages over the traditional one, where all biological replicates (i.e., patients in the same experimental group) are combined in just one set of values. The main advantage of this representation is that we can examine the behavior of the genes independently in each patient, making it possible for us to recognize different behaviors of genes across the patients in the same experimental group. These differences can help us to discover the influence of biological conditions not previously considered in the experiment such as gender or age. Finally, the system provides solutions based on a trade-off of specificity vs. sensitivity, whereas other methods evaluate their solutions over one measure, usually a ratio of False Positives and the total number of genes retrieved [4], [5]. As a result of this trade-off, the procedure provides as output all non-dominated solutions in terms of specificity and sensitivity by application of multiobjective techniques.

The computational procedure we propose solves many of the problems actually present in the process of analyzing a microarray experiment, such as the decision of analytical methodology to follow, extraction of results biologically significant for the experts, proper management of complex experiments harboring experimental conditions, time-series and patients. Therefore, it sets up a robust platform for the analysis of all types of microarray experiments, from the simplest experimental design to the most complex, providing accurate and reliable results.

References

1. Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press.
2. Brown,P. and Botstein,D. (1999) Exploring the new world of the genome with DNA microarrays. Nature Genet., 21 (Suppl.), 33-37.
3. Pan,W., Lin.J. and Le.C. (2001) A mixture model approach to detecting differentially expressed genes with microarray data. Funct. Integr. Genomics, 3(3), 117-124.
4. Li,C. and Wong,W.H. (2003) DNA-Chip Analyzer (dChip). In Parmigiani,G., Garrett,E.S., Irizarry,R. and Zeger,S.L. (eds), The analysis of gene expression data: methods and software. Springer.
5. Tusher,V.G., Tibshirani,R. and Chu,G. (2001) Significance analysis of microarrays applied to the ionizing radiation response. Proc. Natl. Acad. Sci. USA. 98, 5116-5121.

6. Park,T., Yi,S.G., Lee,S., Lee,S.Y., Yoo,D.H., Ahn, J.I. and Lee, Y.S. (2003) Statistical tests for identifying differentially expressed genes in time-course microarray experiments. Bioinformatics, 19(6), 694-703.
7. Der,G. and Everitt,B.S. (2001) Handbook of Statistical Analyses using SAS. Chapman and Hall/CRC.
8. Cheeseman,P. and Oldford,R.W. (1994) Selecting models from data : artificial intelligence and statistics IV. Springer-Verlag, New York.
9. Zwir,I., Shin,D., Kato,A., Nishino,K., Latifi,K., Solomon,F., Hare,J.M., Huang,H. and Groisman,E.A. (2005a) Dissecting the PhoP regulatory network of Escherichia coli and Salmonella enterica. Proc Natl Acad Sci, 102, 2862-2867.
10. Zwir,I., Huang,H. and Groisman,E.A. (2005b) Analysis of Differentially-Regulated Genes within a Regulatory Network by GPS Genome Navigation, Bioinformatics (in press).
11. Chankong,V. and Haimes,Y.Y. (1983) Multiobjective decision making theory and methodology. North-Holland.
12. Deb,K. (2001) Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, Chichester, New York.
13. Agrawal,R., Imielinski,T., Swami,A.N. (1993) Mining association rules between sets of items in large databases. In Buneman, P., Jajodia, S., eds.: Proceedings of the ACM SIGMOD. International Conference on Management of Data, Washington, D.C., 207--216
14. Kooperberg,C., Sipione,S., LeBlanc,M., Strand, A.D., Cattaneo,E. and Olson, J.M. (2002) Evaluating test statistics to select interesting genes in microarray experiments. Hum. Mol. Genet., 11(19), 2223-2232.
15. Mitchell,T. (1997) Machine Learning. McGraw Hill.
16. Duda, R. O., and Hart, P. E. (1973) Pattern Classification and Scene Analysis. John Wiley & Sons, New York, USA.
17. Cordón O, del Jesus, M.J., Herrera, F. (1999) A Proposal on Reasoning Methods in Fuzzy Rule-Based Classification Systems. International Journal of Approximate Reasoning., 20, 21-45.
18. Calvano,S.E., Xiao,W., Richards,D.R., Feliciano,R.M., Baker, H.V., Cho, R.J., Chen, R.O., Brownstein,B.H., Cobb,J.P., Tschoeke,S.K., Miller-Graziano,C., Moldawer,L.L., Mindrinos, M.N., Davis, R.W., Tompkins,R.G. and Lowry,S.F. (2005) The Inflammation and Host Response to Injury Large Scale Collaborative Research Program. A Network-Based Analysis of Systemic Inflammation in Humans. Nature, in press.
19. Tavazoie,S., Hughes,J.D., Campbell,M.J., Cho,R.J. and Church,G.M. (1999) Systematic determination of genetic network architecture, Nat Genet, 22, 281-285.
20. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M. and Sherlock, G. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium, Nat Genet, 25, 25-29.
21. Benitez-Bellon,E., Moreno-Hagelsieb,G. and Collado-Vides,J. (2002) Evaluation of thresholds for the detection of binding sites for regulatory proteins in Escherichia coli K12 DNA. Genome Biol. 3(3)):RESEARCH0013.

Microarray Probe Design Using ϵ-Multi-Objective Evolutionary Algorithms with Thermodynamic Criteria

Soo-Yong Shin, In-Hee Lee, and Byoung-Tak Zhang

Biointelligence Laboratory,
School of Computer Science and Engineering,
Seoul National University, Seoul 151-742, Korea
{syshin, ihlee, btzhang}@bi.snu.ac.kr

Abstract. As DNA microarrays have been widely used for gene expression profiling and other fields, the importance of reliable probe design for microarray has been highlighted. First, the probe design for DNA microarray was formulated as a constrained multi-objective optimization task by investigating the characteristics of probe design. Then the probe set for human paillomavrius (HPV) was found using ϵ-multi-objective evolutionary algorithm with thermodynamic fitness calculation. The evolutionary optimization of probe set showed better results than the commercial microarray probe set made by Biomedlab Co. Korea.

1 Introduction

DNA microarray, especially oligonucleotide array, consists of the DNA sequences called probes, which are DNA complementaries to the genes of interest, on a solid surface. When the molecules of a cell is put to the microarray, if there exists a complementary oligonucleotide to one of the probes, it would hybridize to the probe so that a user can detect it using various methods. In this way, DNA microarray can provide the information on whether a gene is expressed or not for hundreds of genes simultaneously. Therefore, DNA microarray is widely used to study cell cycle, gene expression profiling, and other DNA-related phenomena in a cell; and has become the method of choice to monitor the expression level of a large number of genes.

By the way, microarray depends on the quality of probe sets that used. If a probe hybridizes to not only its target gene but also other genes, the microarray may produce misleading data. Thus, one needs to design the probe set carefully to get precise data. Till now, lots of probe design methods and strategies are suggested reflecting its importance [16]. Gordon and Sensen proposed a Osprey system based on various well-defined criteria [5]. Zuker group implemented OlgioArray 2.0 using thermodynamic data to predict secondary structures and to calculate the specificity of targets on chips [10]. Wang and Seed suggested OligoPicker which uses BLAST search for sequence specificity decision [18].

Though they have shown the good results, the main algorithm of most previous system is a simple *generate and filter-out approach.* Recently, a method based on machine learning algorithms such as naïve Bayes, decision trees, and neural networks has been proposed for aiding probe selection [15]. And in our previous work [8], we used a multi-objective evolutionary algorithm for probe selection of DNA microarray. We designed 19 probes for human papillomaviruses using non-dominated sorting genetic algorithm-II (NSGA-II). In this paper, we improved our previous approach in many ways. First, we reformulated the probe design problem by investigating the characteristics of the probe design. Second, we adopted ϵ-multi-objective evolutionary algorithm (ϵ-MOEA) instead of NSGA-II. In a related field, DNA sequence design for DNA computing, we noticed that ϵ-MOEA outperforms NSGA-II for DNA sequence design problem [12]. Based on these results, we improved the main algorithms to ϵ-MOEA. Third, we changed the fitness criteria of probe design by combining thermodynamic data and sequence similarity search.

In the following sections, we explain the suggested probe design method in detail. In section 2, we briefly introduce the multi-objective optimization problem and formulate the probe design problem as multi-objective optimization problem. Section 3 and 4 describe our probe design method and provide the experimental results. In Section 5, the conclusion will be followed.

2 Multi-Objective Probe Design

2.1 Multi-Objective Optimization Problem

A multi-objective optimization problem (MOP) has a number of conflicting objectives which are to be optimized [1]. For non-conflicting objectives, the optimization of one objective implies the optimization of the other and both objectives can be treated as one objective. And if there exists priority between objectives, one can optimize objectives according to the priority by optimizing single objective which is the weighted sum of objectives. Therefore, for both cases, the given problem becomes a single objective optimization problem. However, in MOP, objectives conflict each other and there is no given priority between objectives, which makes the optimization more difficult than in single objective case.

The general form of multi-objective optimization problem is like the following:

$$\begin{array}{lll} \text{Optimize} & f_m(\mathbf{X}), & m = 1, \cdots, M, \\ \text{subject to} & g_j(\mathbf{X}) \geq 0, & j = 1, \cdots, N, \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, i = 1, \cdots, n & \end{array} \qquad (1)$$

where, $\mathbf{X}$ is a vector of n decision variable $[x_1, \cdots, x_n]^T$, f represents objective, g is constraint, M denotes the number of objectives, and N the number of constraints. $x^{(L)}$ is lower value of decision variable and $x^{(U)}$ is upper value of decision variable.

Given an optimization problem, one's goal is to find optimal solution(s). For a single objective case, the optimality of a solution is determined by simply comparing its objective function value to others. In multi-objective case, the optimality of a solution is determined by domination relation between solutions. A solution X is said to *dominate* other solution Y in the case of maximization when the following two conditions are satisfied and denoted by $X \preceq Y$:

$$\begin{aligned} \forall i \in \{i, \cdots, M\},\ f_i((X)) \geq f_i((Y)), \\ \exists i \in \{i, \cdots, M\},\ f_i((X)) > f_i((Y)). \end{aligned} \tag{2}$$

Therefore, the optimal solutions for a MOP are those that are not dominated by any other solutions. Thus, one's goal in MOP is to find such a *non-dominated set* of solutions.

2.2 Probe Design as Multi-Objective Optimization

There exist several criteria to evaluate the set of probes [5]. We list the generally used conditions for good probes:

1. The probe sequence for each gene should not appear other genes except its target gene.
2. The probe sequence for each gene should be different from each other as much as possible.
3. The non-specific interaction between probe and target should be minimized.
4. The probe sequence for each gene should not have secondary structure such as hairpin.
5. The melting temperatures of the probes should be uniform.

The first three conditions concern with the specificity of the probes. And the secondary structure of a probe can disturb the hybridization with its target gene. Lastly, the probes on a oligonucleotide chip are exposed to the same experimental condition. If the melting temperatures of the probes are not uniform, some probes can not hybridize with its target.

We formulated the above conditions for clear definition of microarray probe design problem. The first condition regarded as a constraint, since it is the basic requirement for probes. And the fifth condition was not considered as one of objectives but was used as the final decision criterion to choose the best solution among diverse Pareto optimal solutions which are the results of the MOEA run.

Therefore, we formulated the microarray probe design using three fitness functions and one constraint. Before going on the formulation of the problem, let us introduce the basic notations. We denote a set of n probes by $P = \{p_1, p_2, \cdots, p_n\}$, where $p_i = \{A, C, G, T\}^l$ for $i = 1, 2, \cdots, n$, l is the length of each probe. And we denote the set of target genes by $T = \{t_1, \cdots, t_n\}$.

The constraint is the basic requirement for probes.

$$g(P) = \sum_{i \neq j} subseq(p_i, t_j), \tag{3}$$

$$subseq(p_i, t_j) = \begin{cases} 1 \text{ if } p_i \text{ occurs in } t_j \text{ at least once} \\ 0 \text{ otherwise} \end{cases}$$

Since the probe sequences should not be the subsequence of the non-target gene sequences (condition 1), this constraint is the basic requirement. And from its definition, this constraint should be zero. Other conditions are implemented as three fitness functions. First one is to prevent hybridization between probe and non-target genes (condition 2). Second is to prevent hybridization between probe and improper position of target genes (condition 3). Even though probe hybridized to the undesired site of its target gene, this can give the right information. Therefore, this seems to be unnecessary fitness functions. However, for more specific probe design, we add this fitness function in our design criteria. Last one is to prohibit forming unwanted secondary structures which can disturb the hybridization between probe and target (condition 4). They could be abstracted as follows:

$$f_1(P) = \sum_{i \neq j} hybridize(p_i, t_j), \tag{4}$$

$$f_2(P) = \sum_{i} hybridize_{target}(p_i, t_i), \tag{5}$$

$$f_3(P) = \sum_{i} secondary(p_i). \tag{6}$$

where, $hybridize(p_i, t_j)$ has non-zero value in proportion to the hybridization likelihood between p_i and t_j. $hybridize_{target}(p_i, t_i)$ is similar with $hybridize(p_i, t_j)$. It increases its value when p_i and t_i hybridize in the non-designed positions which are not the chosen site for p_i in t_i. $secondary(p_i)$ has non-zero value in accordance with the probability that p_i can form the unwanted secondary structures.

The relationship between three objectives are shown in Fig. 1. The graphs were plotted using 4^{20} 20-mer DNA sequences and their Watson-Crick complementary combinations. f_1 or f_2 has the some conflict relation with f_3. Though, in precise, the relation should be treated as random, these objectives could be solved by MOEAs. And f_1 and f_2 has a linear relation as we expected.

From above, the probe design problem is formulated as an MOP with 3 minimization objectives and 1 equality constraints.

$$\begin{aligned} \text{Minimize} \quad & f_i(P), i = 1, 2, 3; \\ \text{subject to} \quad & g(P) \ = 0. \end{aligned} \tag{7}$$

3 Multi-Objective Evolutionary Probe Optimization

To design probe set that satisfies above condition, we used ϵ-multi-objective evolutionary algorithm (ϵ-MOEA). There exist several methods to find such non-dominated set of solutions for a MOP. Among them, evolutionary method is one of the most popular and actively studied methods. It has the advantage that it can provide a set of non-dominated solutions by one run due to a population-based method [1]. And among various multi-objective evolutionary algorithms, ϵ-MOEA has shown the best performance [7, 2, 3].

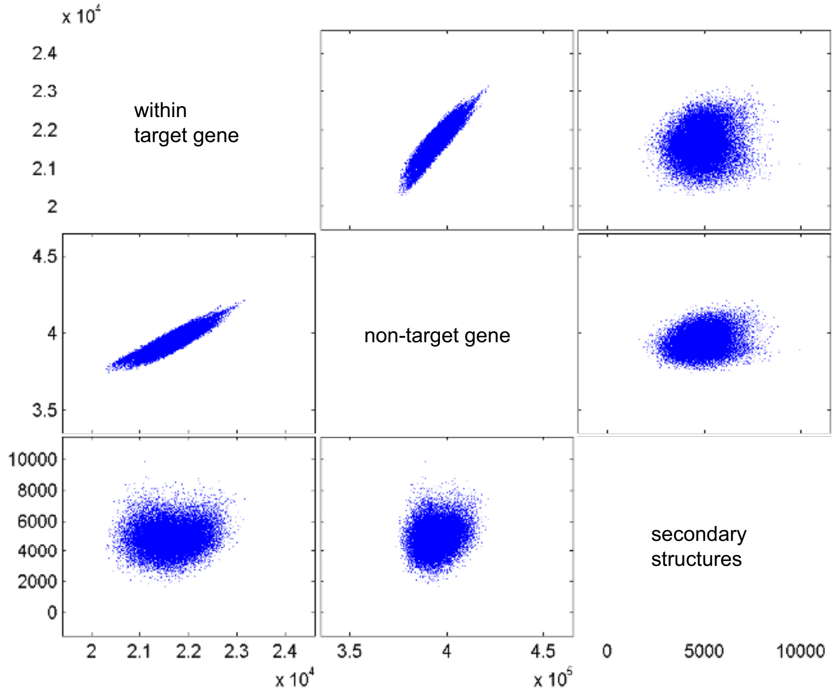


Fig. 1. The relationship between objectives for probe design. The data are generated using 20 mer DNA sequences and Watson-Crick complement.

3.1 ϵ-Multi-Objective Evolutionary Algorithm

ϵ-multi-objective evolutionary algorithm (ϵ-MOEA) is a steady-state genetic algorithm using elite archive and ϵ-dominance relation [7, 3]. The most important characteristic of ϵ-MOEA is the ϵ-dominance relation. In ϵ-dominance relation, x ϵ-dominates y if the difference between x and y is greater than or equal to a certain amount ϵ in all objectives and is strictly better than y by ϵ in at least one objective. The mathematical definition is

$$X \quad \epsilon - dominates \quad Y \quad \Longleftrightarrow \quad (1+\epsilon)f(X) \geq f(Y). \tag{8}$$

The ϵ-dominance is introduced to maintain a representative subset of non-dominated individuals. The ϵ-non-dominated set is smaller than the usual non-dominated set, for the non-dominated solutions which can be ϵ-dominated by others are removed in ϵ-non-dominated set. Therefore, ϵ-Pareto set is a subset of the Pareto-optimal set which ϵ-dominates all Pareto-optimal solutions. And the minimum distance between nearest solutions can be guaranteed by dividing whole search space into many grides. The density of the approximate set can be adjusted by controlling the value of ϵ [7]. Utilizing the ϵ-dominance in selecting representative subset of non-dominated set and maintaining them in the archive

```
1. Randomly generate initial pool.
2. Sort by domination, and set first front as archive.
3. Generate one new individual by choosing the parents from population and archive.
   (a) Choose two individuals from population.
   (b) Choose dominating solution, if dominates; choose random one, otherwise.
   (c) Choose one individuals from archive.
   (d) Perform crossover and mutation.
4. Update archive.
   (a) Replace $\epsilon$-dominated individual(s) in the archive with new individual, if new
       individual $\epsilon$-dominates archive member(s).
   (b) Leave dominating member, if there are more than one archive members in
       the same grid.
   (c) Add new individual, if archive members do not dominate new individual.
5. Update population.
   (a) Replace dominating individual(s) with new individual.
   (b) Replace randomly selected population member with new individual, if there
       is no population member which dominates the new individual.
6. Check termination.
```

Fig. 2. The psedocode of ϵ-MOEA

throughout generations, ϵ-MOEA showed good convergence and diversity performance [2, 3, 7].

The procedure of ϵ-MOEA for probe optimization is explained in Fig. 2. We slightly modified ϵ-MOEA proposed by Deb [2]. At each generation, parents for new offspring are chosen from the population and the archive respectively. The parent from the population is chosen by tournament selection and the parent from the archive is selected randomly. Then, an offspring is produced from these parents and evaluated. The offspring replaces an individual of the population if there exists one dominated by it in usual sense. If the offspring ϵ-dominates one or more members of the archive, it replaces the ϵ-dominated members. Or, the offspring is added to the archive if no archive member ϵ-dominates it and it ϵ-dominates no archive member. Otherwise, the offspring is discarded. Therefore, the ϵ-non-dominated individuals are always the member of the archive. This process is repeated until termination [7].

3.2 Thermodynamic Fitness Calculation

The previous microarray probe design tools can be classified into two groups by their probe specificity evaluation methods: thermodynamic approach [10, 9] and sequence similarity search approach [18, 4]. In thermodynamic approach, the optimum probes are picked based on having free energy for the correct target, and maximizing the difference in free energy to other mismatched target sequences. A sequence similarity search methods used BLAST or BLAT [6] to check cross-hybridization. Since thermodynamic approach is more accurate method between them [10, 9], we calculate the fitness objectives in 2.2 using thermodynamic data. The thermodynamic fitness functions are implemented by the modified Mfold

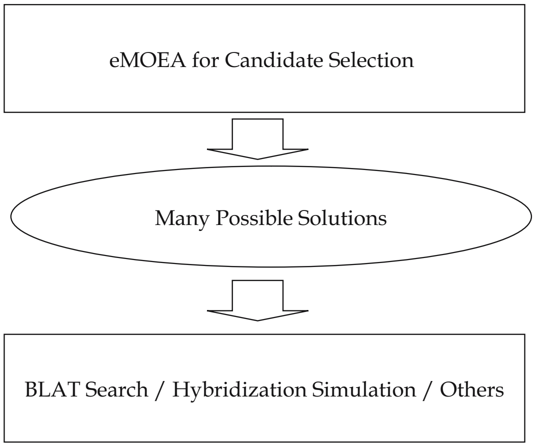

eMOEA for Candidate Selection
Many Possible Solutions
BLAT Search / Hybridization Simulation / Others

Fig. 3. The steps for probe design

[19] for OligoArray problem [10]. We downloaded the stand-alone program source code and slightly modified for fitness functions.

3.3 Probe Selection Procedure

The multi-objective evolutionary algorithm has the advantage that one can get the Pareto optimal solutions at a time. However the users usually need one promising solution, not the set of whole Pareto optimal solutions. Therefore, we incorporated the decision makers to select the most promising solution among Pareto solutions. First, the Pareto optimal solutions can be found by ϵ-MOEA. Then, BLAT search [6], hybridization simulation [13], and melting temperature calculation choose one candidate solution. BLAT is a BLAST-like sequence alignment tool, but much faster than BLAST [6]. NACST/Sim [13] is a hybridization simulation tool to check cross-hybridization based on nearest neighbor model of DNA [11]. Melting temperature is also calculated by nearest neighbor model.

Through these steps, user could be recommended the most promising probe set while maintaining the flexibility to select among various solutions. Using the characteristics of MOEA, we can improve the reliability of the optimized probe set by combining the diverse criteria such thermodynamic fitness calculation, sequence similarity search, and other user-define criteriaThis procedure is summarized in Fig. 3.

4 Experimental Results

4.1 Human Papillomavirus

The proposed constrained multi-objective approach was used to find probe set of human papillomavirus (HPV). HPV is known to be the cause of cervical cancer

[17]. HPV types can be divided into two classes: ones that are very likely to cause the cervical cancer and the others that are not. 19 genotypes of HPV belong to the first class are selected as target genes. The goal is to discriminate each of 19 genotypes among themselves. The selected 19 genes are HPV6, HPV11, HPV16, HPV18, HPV31, HPV33, HPV34, HPV35, HPV39, HPV40, HPV42, HPV44, HPV45, HPV51, HPV52, HPV56, HPV58, HPV59, and HPV66. And to improve the accuracy, L1 region of each gene sequences is chosen. Each gene and L1 region are selected by Biomedlab Co., Korea with experts' laborious works.

4.2 Parameter Settings

Based on the experimental data from Biomedlab, the length of each probe was set to 30 nucleotides long. For ϵ-MOEA, we used the various parameters. The size of population was set as 100 and the maximum generation number as 1, 1,000, 5,000, and 100,000. The crossover and mutation rates were set as 0.9 and 0.01 respectively. The ϵ was set as 1 for better convergence. For BLAT, we use default parameter settings. For NACST/Sim, we set hybridization temperature as 40°C, where the concentration of sodium ion and oligomers were set to $1M$ and $1\mu M$ respectively. The hybridization temperature was decided based on the experimental data from Biomedlab.

4.3 Probe Design Results

Our method is based on evolutionary approach, not a simple generate-filter approach which is used by most previous probe design tools. To check the merits of evolutionary approach, we compared the results by varying maximum generation from 1 to 100,000. Evolutionary algorithm with generation 1 would be the same as generate-filter method. The comparison results are shown in Table 1. As we expected, design with more generation can find better probe set. In the aspect of the number of average cross-hybridization which is checked by NACST/Sim, probe set with more generation produces the less cross-hybridization. A cross-hybridization means the undesirable hybridization between probes and genes. Especially, the comparison result between generation 1 and 1,000 showed the remarkable improvement. This means evolutionary approach can design more reliable probe set compared to the simple method. In addition, more than 1,000 generation did not show the impressive improvement. This result implies one does not need a quite large number of generation to find better probe set.

Table 1. The comparison result for various generation. As generation goes on, the probes show the less cross-hybridizations.

Generation	1	1,000	5,000	100,000
Number of average cross-hybridization	41.33	13.45	13	10.64
Number of Pareto-optimal probe sets	12	11	4	38

Table 2. *in silico* hybridization results for Pareto set with generation 1000

Set	Number of cross-hybridization
0	16
1	15
2	15
3	12
4	23
5	11
6	9
7	10
8	7
9	11
10	19

Table 3. The comparison result between probes in commercial chip (Biomedlab), selected probes using NSGA-II [8], and selected probes with ϵ-MOEA. First row means the undesirable hybridization between probes and genes calculated by NACST/Sim. Second raw represent the similar sequences appear in the wrong position. Therefore, 0 means the probe sequence appear only in its original position. The proposed method (ϵ-MOEA showed best performance for all aspects.

	ϵ-MOEA	NSGA-II	Biomedlab. Probes
# cross-hybridization	7	21	17
BLAT search	0	0 (1 for whole)	0
Melting temperature (^{o}C)	72.58 ± 3.55	74.87 ± 2.34	77.52 ± 5.03

As shown in Table 1, there are various candidate probe sets ($4 \sim 38$) as results of ϵ-MOEA. To choose best probe set among candidate probe sets, we used BLAT with HPV gene sequences first. However, we could not find any cross-hybridization using BLAT unfortunately. Since L1 region of HPV sequences is very well discriminated parts of HPV sequences, there is no similar sequences. Even when we compared L1 region sequences with whole HPV sequences using BLAT, we can find only few similar sequences. Second, we use *in silico* hybridization using NACST/Sim. The results are shown in Table 2. We used NACST/Sim for Pareto set found by 1000 generation. As explained previously, 1000 generation showed the most significant result and other runs required too much run times. As a result, we chose set no. 8 for final probe set, since that set showed the smallest number of cross-hybridization.

To verify the reliability of final probe set, we compared the probe set by ϵ-MOEA with the probes in commercial chip made by Biomedlab and the probe set by NSGA-II [8]. Table 3 showed the comparison results. ϵ-MOEA found the best probe set. Probe set by NSGA-II has three times more cross-hybridizations and Biomedlab probes has 2.5 times more cross-hybridizations. We ran BLAT for L1 region and whole HPV sequence respectively. BLAT found one similar sequences for whole HPV sequences in NSGA-II probe set, and could not find any more

Table 4. The final set of probes chosen by the proposed method for HPV

HPV Type	Probe Sequence
HPV6	CATGTACTCTTTATAATCAGAATTGGTGTA
HPV11	TCTGAATTAGTGTATGTAGCAGATTTAGAC
HPV16	TCCTTAAAGTTAGTATTTTTATATGTAGTT
HPV18	ATGTCTGCTATACTGCTTAAATTTGGTAGC
HPV31	CTTAAATACTCTTTAAAATTACTACTTTTA
HPV33	CTGTCACTAGTTACTTGTGTGCATAAAGTC
HPV34	GTGCAGTTGTACTTGTGGATTGTGTACCTA
HPV35	TTTATATGTACTGTCACTAGAAGACACAGC
HPV39	AAGGTATGGAAGACTCTATAGAGGTAGATA
HPV40	CTTGAAATTACTGTTATTATATGGGGTTGG
HPV42	AAATTAGCAGCTGTATATGTATCACCAGAT
HPV44	TTGCTTATATTGTTCACTAGTATATGTAGA
HPV45	CATGTCTACTATACTGCTTAAACTTAGTAG
HPV51	AAAGTTACTTGGAGTAAATGTTGGGGAAAC
HPV52	TTTATATGTGCTTTCCTTTTTAACCTCAGC
HPV56	ATTAATTTTTCGTGCATCATATTTACTTAA
HPV58	TTTATATGTACCTTCCTTAGTTACTTCAGT
HPV59	TAGGTGTGTATACATTAGGAATAGAAGAAG
HPV66	GAAGGTATTGATTGATTTCACGGGCATCAT

similar sequences. Probe set by ϵ-MOEA has also the lowest melting temperature among three probe sets. Though NSGA-II has the smallest melting temperature variation, the difference is not so significant compared to ϵ-MOEA. The reason why NSGA-II found the near uniform melting temperature probe set is NSGA-II used the melting temperature variation as one of objectives [8]. Even though we did not use that objective, our approach can find the comparable results. The probes practically used in Biomedlab showed the poorest results in melting temperature, even though the melting temperature variation is important for the microarray experiment protocols. The final probe set generated by the proposed approach is shown in Table 4.

5 Conclusion

We formulated the probe design problem as a constrained multi-objective optimization problem and presented a multi-objective evolutionary method for the problem. Because our method is based on multi-objective evolutionary algorithm, it has the advantage to provide multiple choices to users. And to make it easy to choose among candidates, we suggested the criteria as an assistant to the decision maker. It is shown that the proposed method could be useful to design good probes by applying it to real-world problem and comparing them to currently used probes.

Though the previous works focused on finding the moderate probe set in short time, we focused on improving the quality of probe set. Therefore, our approach need more computational time compared to the previous approaches. However, we showed the small iterations can improve the probe set quality significantly. In addition, MOEA can combine thermodynamic methods and sequence similarity search. Since these results are the preliminary results, it is necessary to optimize several time consuming stages.

Acknowledgements

This research was supported in part by the Ministry of Education & Human Resources Development under the BK21-IT Program, the Ministry of Commerce, Industry and Energy through MEC project, and the NRL Program from Korean Ministry of Science and Technology. The RIACT at Seoul National University provides research facilities for this study. The target gene and probe sequences are supplied by Biomedlab Co., Korea.

References

1. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, Ltd., 2001.
2. K. Deb, M. Mohan, and S. Mishra. A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. KanGAL Report 2003002, Kanpur Genetic Algorithm Laboratory, Indian Institute of Technology Kanpur, 2003.
3. K. Deb, M. Mohan, and S. Mishra. Towards a quick computation of well-spread Pareto-optimal solutions. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization*, pages 222–236, 2003.
4. K. Flikka, F. Yadetie, A. Laegreid, and I. Jonassen. Xhm: A system for detection of potential cross hybridizations in dna microarrays. *BMC Bioinformatics*, 5(117), 2004.
5. P. M. K. Gordon and C. W. Sensen. Osprey: a comprehensive tool employing novel methods for design of oligonecleotides for dna sequencing and microarrays. *Nucleic Acid Research*, 32(17):e133, 2004.
6. W. J. Kent. BLAT–the BLAST-like alignment tool. *Genome Research*, 12(4):656–664, 2002.
7. M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimizatin. *Evolutionary Computation*, 10(3):263–282, 2002.
8. I.-H. Lee, S. Kim, and B.-T. Zhang. Multi-objective evolutionary probe design based on thermodynamic criteria for HPV detection. *Lecture Notes in Computer Science*, 3157:742–750, 2004.
9. F. Li and G. D. Stormo. Selection of optimal DNA oligos for gene expression arrays. *Bioinformatics*, 17:1067–1076, 2001.
10. J.-M. Rouillard, M. Zuker, , and E. Gulari. OligoArray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach. *Nucleic Acids Research*, 31(12):3057–3062, 2003.
11. J. SantaLucia Jr. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proceedings of the National Academy of Science of the United States of America*, 95:1460–1465, 1998.

12. S.-Y. Shin. *Multi-Objective Evolutionary Optimization of DNA Sequences for Molecular Computing*. PhD thesis, School of Computer Science and Engineering, Seoul National University, Seoul, Korea, 2005.
13. S.-Y. Shin, H.-Y. Jang, M.-H. Tak, and B.-T. Zhang. Simulation of DNA hybridization chain reaction based on thermodynamics and artificial chemistry. In *Preliminary Proceedings of 9th International Meeting on DNA Based Computer*, page 451, 2004.
14. S.-Y. Shin, I.-H. Lee, D. Kim, and B.-T. Zhang. Multi-objective evolutionary optimization of DNA sequences for reliable DNA computing. *IEEE Transactions on Evolutionary Computation*, 9(2):143–158, 2005.
15. J. B. Tobler, M. N. Molla, E. F. Nuwaysir, R. D. Green, and J. W. Shavlik. Evaluating machine learning approaches for aiding probe selection for gene-expression arrays. *Bioinformatics*, 18:164–171, 2002.
16. S. Tomiuk and K. Hofmann. Microarray probe selection strategies. *Briefings in Bioinformatics*, 2(4):329–340, 2001.
17. J. M. M. Walboomers, M. V. Jacobs, M. M. Manos, F. X. Bosch, J. A. Kummer, K. V. Shah, P. J. F. Snijders, J. Peto, C. J. L. M. Meijer, and N. Munoz. Human papillomavirus is a neccsary cause of invasive cervical cancer worldwide. *Journal of Pathology*, 189(1):12–19, 1999.
18. X. Wang and B. Seed. Selection of oligonucleotide probes for protein coding sequences. *Bioinformatics*, 19(7):796–802, 2003.
19. M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406–3415, 2003.

An Algorithm for the Automated Verification of DNA Supercontig Assemblies

Nikola Stojanovic

Department of Computer Science and Engineering,
The University of Texas at Arlington, Arlington, TX 76019, USA
Phone 817-272-7627, FAX 817-272-3784
nick@cse.uta.edu

Abstract. Genome sequencing has achieved tremendous progress over the last few years. However, along with the speedup of the process and an ever increasing volume of data there are continuing concerns about the quality of the assembled sequence. Many genomes have been sequenced only to a draft, leaving the data in a series of more–or–less organized scaffolds, and many feature a small, but not negligible number of misassembled pieces. In this paper we present a new method for automated flagging of potential trouble spots in large assembled supercontigs. It can be incorporated into existing quality control pipelines and lead to a considerable improvement in the sensitivity to certain types of errors.

1 Introduction

Despite the advances in the genome sequencing technology, quality of the assembled products remains a major concern. A recent study done by the Genome Sciences Centre in Vancouver, Canada, in collaboration with several sequencing centers in the United States has identified an average of 4.19 to 4.57 assembly problems per one million bases including an average of 0.3 to 0.4 wrong and misassembled clones (Rene Warren, personal communication). While these numbers are reasonably low for such complex operation as the assembly of vertebrate genomes, they are sufficiently high to warrant continued attention.

The first vertebrate genome assembled was that of human. This task was preceded by the construction of detailed maps and the establishment of a large number of markers along chromosomes [5]. Only after this task has been substantially completed the sequencing of many large insert clones (Yeast Artificial Chromosomes, YACs, at first, followed by more stable Bacterial Artificial Chromosomes, BACs) could begin. At first, the Human Genome Project centers intended to perform the sequencing in a structured way, following the maps and progressively expanding the tiling path of large insert clones (further referred to as LICs), finishing them to full accuracy (initially set to less than one error in every ten thousand bases) in the process. While the groups outside the United States continued pursuing this strategy, which resulted in the early completion of chromosomes 21 [2] and 22[1], the emergence of the whole–genome shotgun strategy [15] and subsequent challenge to the public effort by a private company, Celera Genomics, led to the change of course for the consortium members located in the US. The human genome has been first released as draft sequence [6] covering about 90% of its euchromatic part, mostly as a collection of unordered and unoriented contigs featuring 147,821 gaps. The finished sequence followed almost three years later [7].

While Celera constructed its scaffolds of the human genome using the whole–genome shotgun approach and mixing clones of different lengths [14], the public HGP was completed using LICs, mostly BACs from CalTech and RPC-11 libraries. Initially there were about 600,000 of these clones, of average length of 150Kb, and only a part of these were selected for further breaking into sequencing clones (M13 or plasmid, each of about 2Kb in length). The remainder has been end sequenced, generating paired reads of about 500 to 1,000 bases long, and used as an aid in the final assembly of chromosomes. BAC clones have been created by partial digestion of DNA with restriction enzymes, EcoRI and HindIII at CalTech, and EcoRI and MboI for RPC-11. In order to generate the desired range of LIC sizes the enzymes have been appropriately diluted.

After the HGP switch to the rapid generation of draft sequence the goals of clone selection have changed from the orderly generation of tiling paths of finished LICs to the identification of non-overlapping clones, in order to assure the production of the maximal possible amount of new sequence (Ken Dewar, personal communication). In consequence, after generating the draft the public consortium was left with a daunting task of organizing thousands of contigs in which more than 50% of the bases lied in assembled regions of less than 100Kb. This process was prone to laboratory errors in clone handling, which we have addressed in an earlier paper [13], and to the incorrect placement of LICs in tiling paths. In this manuscript we address a new computational method whose original development was done in order to address the latter issue.

The efforts of generating the complete human sequence could be broadly classified in two categories: finishing of individual LICs and their arrangement along chromosomes. While the challenges concerning the former mostly lied in laboratory work, the latter primarily involved computation. The first arrangement have been produced by W. James Kent at the University of California, Santa Cruz [9], who used the information contained in the initial sequence contigs, linkage and fingerprint maps, mRNA and Expressed Sequence Tags (EST) data, and BAC end sequences. The Institute for Genomic Research (TIGR) in Rockville, Maryland, provided end sequences for about 500,000 BACs from the human libraries, out of which about 300,000 were sequenced from both ends (generating around 600,000 paired reads) and the remaining 200,000 were unpaired. In addition, about 750,000 fosmid clones (similar to BACs, but much shorter — about 40Kb in size, on average) have been created and end–sequenced for the verification of the assembly. These clones provided another 8× coverage of the human genome, bringing the total to almost 30–fold redundancy [7].

After the completion of the Human Genome Project, the sequencing community has been steadily moving towards the whole–genome shotgun assembly method. The mouse [10] and rat [12] genomes have been assembled using a hy-

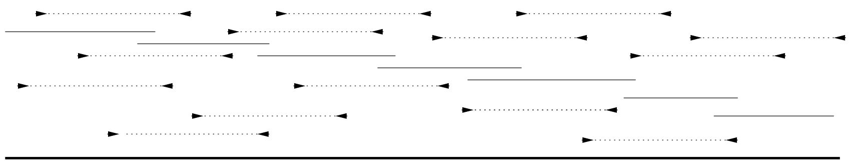


Fig. 1. Coverage of genomic sequence by large insert clones. These actually sequenced, represented by solid lines form a tiling path, while these only end-sequenced (represented by solid ends with reads pointing towards each other) are shown as dotted lines. If the paired end–reads were placed on the path in the right orientation and at about right distance, this was considered as additional coverage of the enclosed bases. The thick line at the bottom represents the assembled chromosomal region.

brid approach, and the subsequent vertebrate genomes were assembled by new mega–assemblers [8, 11, 4]. However, large insert clones still play a role in the assembly of genomic scaffolds [3, 14], which use both BAC and fosmid end reads in addition to shorter fragments.

2 Distribution of Large Insert Clones in the Genome

The finished human genome, and the other ones subsequently sequenced, have been verified for correctness using several methods, however an early signal that there might be a problem with an assembled scaffold comes from the depth of its coverage. At the level of LIC assembly this would be the coverage by BACs and similar clones, and the method used by the sequencing centers involved the detection of areas where the coverage differed from the expected by more than 3.5 standard deviations (σ). While it is almost certain that coverage deviating more than 3.5σ (and thus less than 0.0005 likely to occur by chance) indicates an error, this criterion may miss quite a few better hidden problems. More reasonable boundary would be at 99% or even 95% significance (about two standard deviations from the mean), however the number of clones flagged by such screen would be very large, especially in the light of the properties of LIC coverage.

The development of the method described in this paper originated in the late days of the HGP. The genome closure group at the Whitehead Institute Center for Genome Research (further referred to as WICGR) was in charge of finishing human chromosomes 8, 11p, 15, 17 and 18q. The selection of clones and the subsequent verification of the correctness of the assembly were done through hybridizations done at the laboratory bench, comparison of finished sequences with maps and checking of the placement of BAC end reads. Since only a subset of available human BACs have been actually sequenced, the consistent placement of ends of unsequenced clones, at the right distance and orientation, provided additional virtual coverage and helped ensure that the assembly was correct. The placement of such clones is illustrated in Figure 1.

A large fraction of BAC end reads (paired and unpaired) provided by TIGR have not been mapped on the draft assembly of the human genome, due to its

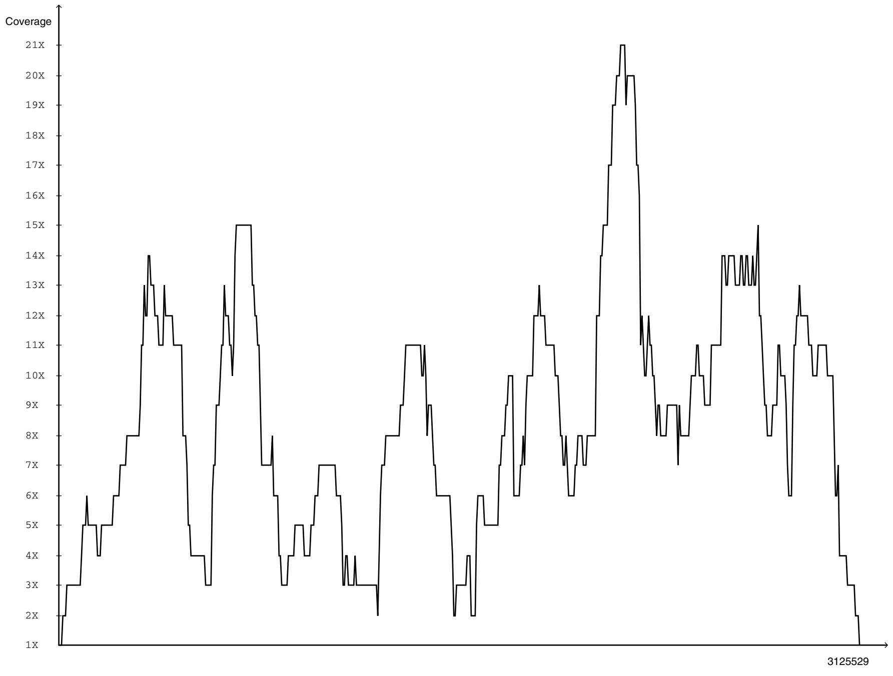


Fig. 2. Coverage of RefSeq supercontig NT_000765 with BAC clones. Positions within the supercontig are plotted on the X-axis (0–3173457). Fold coverage over the sampled positions is represented along the Y-axis.

greatly repetitive structure and the fact that TIGR has used the draft masked for known repeats. Using unmasked sequence, the WICGR group succeeded to place about 25% more reads, and assure LIC coverage of almost 15× throughout the part of the genome it was in charge of, not counting the additional 8× from fosmid libraries (Nathaniel Strauss, WICGR closure group, personal communication). The expectation was that every DNA base would be covered by a relatively stable number of clones, roughly around the mean, and a missassembly would be indicated by anomalies. However, it was somewhat surprising to see that the actual coverage could show large variations, as illustrated in Figure 2.

The first suspect for this apparent paradox was an uneven distribution of the restriction enzyme target sites in parts of the genome. While this is generally true, in particular for heterochromatic and other satellite regions, in most chromosomal DNA these sites are distributed in agreement with Poisson expectation. However, although random, the particular placement layout in any sequence dictates a certain coverage pattern, with well defined positions of unusually high or unusually low coverage, i.e. the number of LICs selected from that region. This has been verified by simulating the clone library construction *in silico* and selecting the number of clones to provide several hundred, and even several thousand–fold virtual coverage of the target regions. Even at numbers

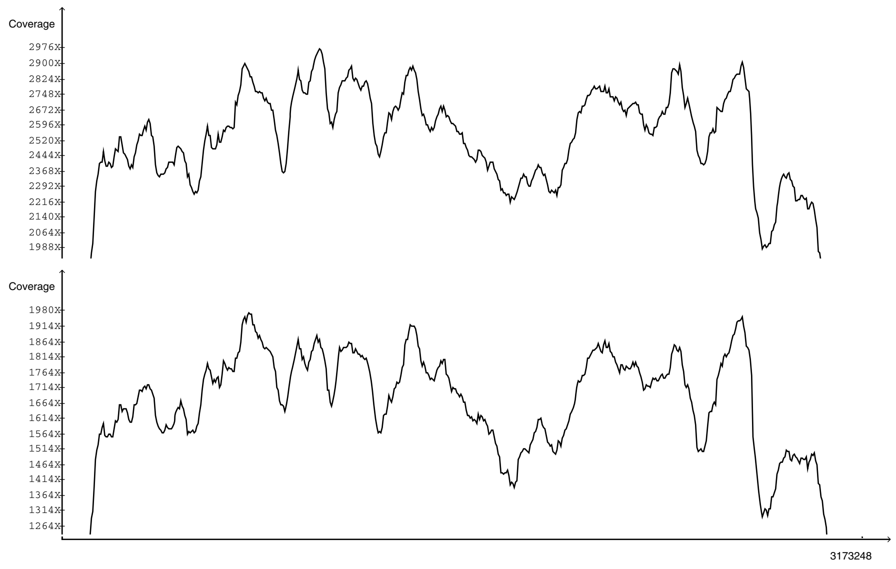


Fig. 3. Coverage curves for a 3Mb genomic region. The top part of the figure plots the curve achieved at virtual coverage up to 3000× (average ∼ 2500×). The bottom part shows the coverage up to 2000× (average ∼ 1600×). At such high coverages the peaks and dips of the curve tend to stabilize at fixed locations, although minor variations can still be detected.

as low as 100×, the limiting curve (to which we shall further refer by $\mathcal{L}$) would converge to a pattern characteristic for that region, with well defined peaks and bottoms, as illustrated in Figure 3.

This observation led to an idea to compare the actual coverage of a genomic region not with an *a priori* determined mean coverage, but with its own characteristic limiting curve $\mathcal{L}$. Given a long supercontig, our software would be trained to learn $\mathcal{L}$, then compare the actual coverage (whose curve will be referred to by $\mathcal{C}$, or $\mathcal{C}_k$ for $k\times$ coverage) with it. If $\mathcal{C}$ would feature peaks and dips at the positions consistent with $\mathcal{L}$ that would indicate a correct assembly almost regardless of the number of mapped clones.

3 Coverage Simulation and Comparison Algorithm

Although $\mathcal{L}$ stabilizes at high values, the practical coverage redundancies are usually low, and for what amounts to a small sample one can expect considerable random variation. We have thus decided to apply our algorithm only to outliers showing a difference from the mean greater than two standard deviations.

Our algorithm starts by reading the sequence of the assembled supercontig, and a file describing the conditions under which the clone libraries covering this region have been created. This file contains the percentages of the LICs created by each restriction enzyme, as well as the enzyme target site, percent dilution (for partial digest) and the permissible clone size range. In addition, this file can contain the information about the other clones used (fosmids or plasmids), whether they are digested by restriction enzymes or randomly sheared, and what percentage of each has been included in the libraries. In the training phase, the software mimics the process done in the laboratory, constructs the clone libraries covering the segment and outputs the file containing the virtual libraries.

The creation of the virtual libraries is the most time–consuming step of the analysis. In order to faithfully reproduce the laboratory procedure, the software must search the sequence for the target sites of the restriction enzymes (GAATTC for EcoRI, AAGCTT for HindIII and GATC for MboI, for instance). It does not need to scan both strands, since these enzymes cut at palindromic sequences, but it needs to make a random decision whether to cut or not every time a site is found, in accordance with the specified dilution. Thus, for instance, a 2.5% dilution of a six–cutter enzyme would dictate a cut with only 0.025 probability — since under the assumption of equal nucleotide representations in the genome the likelihood of finding a target is $\frac{1}{4^6}$, the probability of a cut at any particular position would be about $p \approx 6 \times 10^{-6}$, i.e. about every 164Kb. Since the occurrence of the cuts is a Poisson process, one can find the probability of the next cut within the permissible clone size range using the exponential distribution, giving $\mathrm{P}\{a \leq X \leq b\} = e^{-pa} - e^{-pb}$. If $a = 120,000$ and $b = 180,000$ this would be $\mathrm{P}\{120{,}000 \leq X \leq 180{,}000\} \approx 0.15$. Virtual clones whose size falls outside of this range must be discarded. On a Unix workstation this may take several minutes per megabase, for higher coverages, so we have limited it to 500× or less in practical runs. Consequently, the scanning of the entire human genome would take several days on a single workstation, and several hours on a supercomputing system such as the UTA Distributed and Parallel Computing Cluster.

After the virtual LIC library covering the sequence in the input has been constructed, another module takes it over to map the ends of these clones to the right positions. This step is not necessary when constructing $\mathcal{L}$, but it is essential for testing. The clone ends mapping introduces errors at rates specified as parameters, including the percentage of clones for which only one end would be sequenced (thus mimicking the discarding of poor quality reads during the actual sequencing), and the percentage of clones where one or both ends cannot be unambiguously mapped to the genome. By manipulating these parameters it is possible to test the behavior of the software under various scenarios.

The most common error during the construction of large supercontigs, spanning millions of base pairs, is in the collapsing of regions of large segmental duplications. If the sequences of two copies are very similar (so that the differences can be attributed to genetic variation between the individuals whose DNA has been used for the libraries, or, in a very small number of cases, sequencing errors), the assembly may lay two copies on the top of each other, causing the omission of DNA between the duplicate loci. As shown in Figure 4, in terms of the coverage of the region by clones, such situation may lead to either reduced or

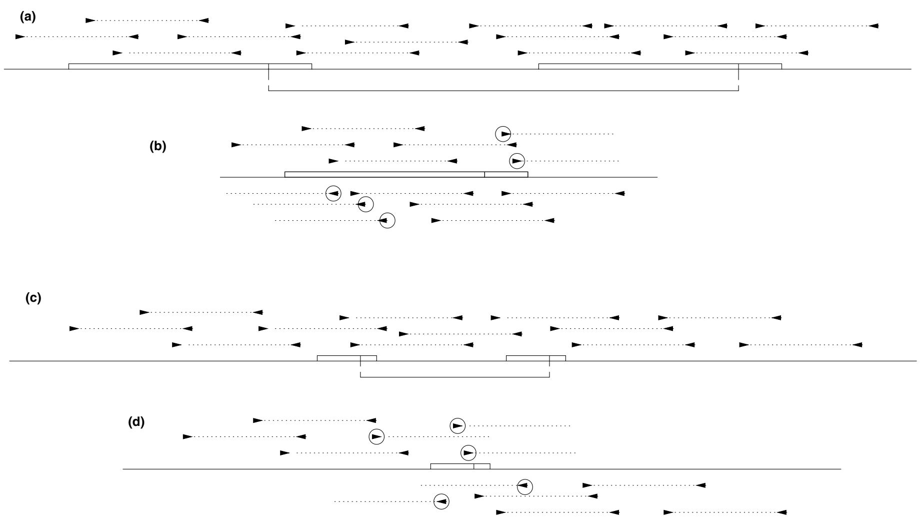


Fig. 4. Two possible interleaving sequence deletion scenarios: (a) Line represents chromosomal DNA, with two narrow boxes indicating a large (> 200Kb) segmental duplication. LICs covering the area are shown on the top. Bottom line connects spots where an incorrect link has been established. The area between the duplicates is presumed to be longer than a single clone; (b) Collapsed segments with the area between them deleted. Spanning clones from the left copy are shown at the top, and these from the right copy at the bottom. Clone end sequences whose matching other end now fails to map in the region are circled; (c) On a line representing chromosomal DNA, two narrow boxes indicate shorter (< 100Kb) duplicated segment. LICs covering the area are shown on the top. Bottom line connects spots where an incorrect link has been established. The area between the duplicated segments is presumed to be longer than a single clone; (d) Collapsed segments with the area between them deleted. Spanning clones from the left copy are shown at the top, and these from the right copy at the bottom. Clone end sequences whose matching other end now fails to map in the region are circled. Both deletions are characterized by anomalous coverage, with spikes in the number of unpaired clone ends along the edges of the collapsed duplications.

increased coverage in conjunction with the increase in the number of errors, i.e. clone ends mapping to the region at unlikely distance, or as unpaired matches. In both cases it is unlikely that the limiting curve $\mathcal{L}$ for the region would confirm such spike, and indeed for suitable lengths of duplicated sequences (see the results below) we have not seen a case where a combination of standard deviation measure and the comparison of $\mathcal{L}$ and $\mathcal{C}$ has not identified the problem spot.

The core of our method is the comparison module. It uses the constructed $\mathcal{L}$ and scans the region (supercontig) in sliding windows of pre-set size, which can be adjusted in accordance with the conditions of the assembly. In our runs we have used windows of size 100Kb, since we were looking primarily at the deletions due to missassemblies at the LIC level. If the length d of the duplicated area is less than one clone size L and k is the overall redundancy of coverage, then the coverage over the collapsed duplicates would be reduced to about $\frac{kd}{L}$ with areas up to L in length on each side of the collapsed duplicates with error rates increased above the background to approximately $\frac{k(L-d)}{L}$. If $d > L$ then the collapsed area would feature an increase in coverage of about $\min(2k, \frac{kd}{L})$ over the midpoint of the collapsed segments with coverage gradually falling to k over the $\min(L, \frac{d}{2})$ bases on both flanks of the collapsed region in addition to the error rate of $\sim k$ for up to L bases around the flanks. In all cases, the signs of trouble should be present over at least 100Kb. This number would be different depending on the mixture of LIC sizes used in different sequencing projects.

The calculation of the mean expected coverage $\mu_{\mathcal{L}}$ and the standard deviation $\sigma_{\mathcal{L}}$ for $\mathcal{L}$ and $\mu_{\mathcal{C}}$ and $\sigma_{\mathcal{C}}$ for $\mathcal{C}$ are done at the supercontig level, but the comparisons are done locally in each window. The number of sampling points n within a window is automatically determined based on the actual coverage k of the examined region, and set to 1.5 the expected number of clone starts and ends. If, for instance, the coverage is $20\times$ in clones of 150Kb, it is expected that a region of 100Kb would feature about 27 points where the coverage changes (a new clone starting or an old one ending), so we would chose 40 sampling points. However, the coverage values for the limiting curve (expressed in hundreds, if not thousands) and the actual data ($10\times$ – $30\times$) need to be put on a uniform scale. If the clones layout were completely random, then the amount of coverage over any point would be normally distributed, so we convert both the limiting and actual values at sampling points x_i (we denote them by $\mathcal{L}(x_i)$ and $\mathcal{C}_k(x_i)$) to the standard normal curve as $z_i^{\mathcal{L}} = \frac{\mathcal{L}(x_i)-\mu_{\mathcal{L}}}{\sigma_{\mathcal{L}}}$ and $z_i^{\mathcal{C}_k} = \frac{\mathcal{C}_k(x_i)-\mu_{\mathcal{C}_k}}{\sigma_{\mathcal{C}_k}}$. We then compare $z_i^{\mathcal{L}}$ and $z_i^{\mathcal{C}_k}$ — if both are within two standard deviations from the 0 mean (indicating that a deviation was neither expected nor has happened) for all $i = 1, n$ we accept the coverage over the window as correct. However, if this condition is not satisfied for any point x_i additional constraints are examined:

1. It must be that $\mid z_i^{\mathcal{L}} - z_i^{\mathcal{C}_k} \mid < 3.5 \quad \forall i \in [1, n]$, and
2. The Pearson correlation coefficient r calculated over all $z_i^{\mathcal{L}}$ and $z_i^{\mathcal{C}_k}$ must not reject non–correlation of $\mathcal{L}$ and $\mathcal{C}_k$ using $\frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$ as a Student–t variable with $n-2$ degrees of freedom, at 95% significance.

Only if both 1 and 2 above are satisfied the window is considered correct, otherwise it is reported as a potential problem spot.

4 Algorithm Performance

Even after many adjustments of our software it was rejecting the assembly of the RefSeq supercontig NT_000765, whose coverage is shown in Figure 2, as incorrect at two loci, around 1.2Mb and around 2.3Mb. Further examination of that supercontig has indeed established a missassembly, and NT_000765 has been subsequently withdrawn from the GenBank. However, although this software has been used on several genomic regions, it has not been incorporated into the

Table 1. The results of the analysis of 27 windows within a 3Mb genomic region under various simulated coverages. Each simulation has been repeated 10 times with error rates ranging between 0% and 10%. For each coverage the comparison has been made with the limiting curve $\mathcal{L}$ constructed from virtual coverage of 500×.

Coverage	Total windows	Windows outside $\pm 2\sigma$	Percentage outside $\pm 2\sigma$	Windows flagged problematic	False positive percentage
300×	270	14	5.19%	0	0%
100×	270	28	10.37%	2	0.74%
30×	270	48	17.78%	24	8.89%
20×	270	44	16.3%	30	11.11%
10×	270	52	19.26%	34	12.59%

genome closure pipeline (due to the departure from WICGR and HGP of both the author and the closure group leader[1] who requested this work).

In order to gain a systematic perspective on the performance of our algorithm we have done a series of simulations using several supercontigs of 2Mb to 5Mb, downloaded from the RefSeq division of the GenBank. Actual coverages have been simulated at various levels, and the BAC ends mapping error rate has been varied between 0 and 35%. The results obtained at $< 10\%$ error rate on a 3Mb long sequence are shown in Table 1. As it can be seen from the table, the number of widows failing the two standard deviations test at coverage 300× is about 5%, which corresponds well with the number of windows expected to have coverage outside $\pm 2\sigma$ bound, by chance. Since at 300× the coverage curve is mostly stable, it indicates that the number of outliers is not unusual, and that it is consistent with the Poisson distribution of the recognition sites for the restriction enzymes. However, the particular locations of these outliers are characteristic of the genomic region in question, as demonstrated by excellent correlation of $\mathcal{C}_{300}$ curve with $\mathcal{L}$. At $\mathcal{C}_{100}$ and further down to more practical coverages of $\mathcal{C}_{30}$, $\mathcal{C}_{20}$ and $\mathcal{C}_{10}$ the number of violations becomes progressively larger, up to almost 20%. This is partially because the chance outliers are more dispersed at smaller sample sizes, and for a window to fail the $\pm 2\sigma$ test it is enough that one of its sample points fall outside these bounds, in either $\mathcal{L}$ or $\mathcal{C}$. In these cases, the number of outliers is still higher than expected, perhaps due to the difference in global versus local variance.

At low coverages the false positive ratio can be high (up to 12.59% of windows for 10× at error rate $\leq 10\%$, and higher in the presence of more errors, when it starts behaving as a random sequence — Table 2), but it still cuts the number of windows that need further checking to about half of these failing the $\pm 2\sigma$ test. As mentioned above, the two standard deviations threshold is much better than 3.5 used in the HGP, and our software has still successfully flagged every missassembled region of the right size which we were aware of (and, in particular, the errors deliberately introduced for testing).

[1] Dr. Ken Dewar, now at McGill University and Genome Quebec Innovation Centre in Montreal, Canada.

For LIC sizes of ∼150Kb, when the segmental duplications are over less than 100Kb, the coverage at the collapsed area would concentrate around the mean of $\frac{2}{3}k$ or less and when the duplicated areas are longer than 200Kb it would peak at around $\frac{4}{3}k$ or more, up to $2k$. In both cases our software was successful in identifying the introduced problem spots, due to a low probability of correlating outliers in both $\mathcal{L}$ and $\mathcal{C}$. This does not mean that our algorithm has zero probability of a false negative, only that in these cases its false negative ratio is low and that it has not happened in our tests. However, when the duplicated area is between 100Kb and 200Kb the expected coverage over the collapsed part is about the same as normal, so the algorithm is more vulnerable to errors. In fact, its incorrect assumption of correlation between $\mathcal{L}$ and $\mathcal{C}$ is similar to when $\mathcal{C}$ is constructed on a sequence different than that used to construct $\mathcal{L}$. The results of the comparisons of unrelated $\mathcal{L}$ and $\mathcal{C}$ are shown in Table 2.

Table 2. The results of the analysis of 27 windows within two unrelated 3Mb genomic regions under various simulated coverages. Each simulation has been repeated 10 times with no introduced errors. For each coverage the comparison has been made with the unrelated limiting curve $\mathcal{L}$ constructed from virtual coverage of 500×.

Coverage	Total windows	Windows outside $\pm 2\sigma$	Percentage outside $\pm 2\sigma$	Windows flagged problematic	Problematic percentage
300×	270	97	35.93%	70	25.93%
100×	270	69	25.56%	50	18.52%
30×	270	90	33.33%	65	24.07%
20×	270	81	30.0%	60	22.22%
10×	270	80	29.63%	61	22.59%

From Table 2 it can be seen that many windows never get into testing for the correlation with $\mathcal{L}$. Since both $\mathcal{L}$ and $\mathcal{C}$ are correctly constructed for their respective regions, there is a large proportion of windows in which no points violate the $\pm 2\sigma$ rule. However, since $\mathcal{L}$ and $\mathcal{C}$ are not related, their outliers are uncorrelated, and thus a larger percentage of their windows (25.56–35.93 versus normal 5.19–19.26) has a $\pm 2\sigma$ outlier in either $\mathcal{L}$ or $\mathcal{C}$. Because of the random arrangement of these outliers the decrease in coverage for $\mathcal{C}$ has only marginal effect on their number. While more than two thirds of the outliers have been identified as problematic, there was still a chance of a sufficient correlation between $\mathcal{L}$ and $\mathcal{C}$, leading to a considerable false negative rate.

In consequence, while this algorithm performs reasonably well for the missassemblies resulting from collapsing duplications less than $\frac{2L}{3}$ and greater than $\frac{4L}{3}$ (although with a substantial false positive ratio), it is not appropriate for detecting these of about $\frac{2L}{3}$ through $\frac{4L}{3}$. These should be checked for by other methods, but the task is now easier as the approximate size of the duplications can be targeted.

5 Discussion

No single sequence assembly quality check works best, and a sequencing facility should apply multiple methods in order to assure the correctness of their data. In particular, when the duplicated genome sequences are adjacent, the analysis should rely on the sizes of clones and the orientation of their end reads in addition to the techniques described in this paper. However, the algorithm we have described provides a considerable improvement in sensitivity when compared with the simple 3.5σ test, and reduces the need for checking the outliers of two standard deviations for about 50%. Most of the regions which are labeled suspicious by our software can be relatively quickly checked for the presence of flanking spikes in the error rates, thus reducing the need for more detailed examination to only a handful of serious suspects.

A new tool, SEMBLANCE, is currently under development at the Washington University Genome Sequencing Center (David Messina, personal communication). This software is designed for the comprehensive assessment of the quality of whole–genome sequence assemblies, assessment of the impact of physical maps on the assembly quality and the comparison of assemblies done at different levels of coverage redundancy. So far, SEMBLANCE has been applied to the analysis of whole–genome assemblies of the chimpanzee genome, at very low coverages, comparing them with chimpanzee BAC clones not used in the assemblies. Since the whole–genome strategies generally include a significant proportion of LICs (BACs and fosmids) the algorithm we have described here would be useful as a part of a quality control toolkit such as SEMBLANCE.

Shearing of the DNA, as opposed to the digestion by restriction enzymes, has been the method of choice for the construction of small sequencing clones for a long time, and recently the genomic library construction efforts have moved towards the application of this technology to fosmids, as well. Once the assemblies start being done based exclusively on sheared clones, we expect that the algorithm described here would lose much of its relevance — since the boundaries of sheared clones are not associated with any particular sequence motif and are theoretically uniformly distributed throughout the genome, one can expect that $\mathcal{L}$ would be flat, and that no particular "signature" limiting curve could be associated with a supercontig. However, many already assembled genomes still need to be refined, and some even partially reassembled, and many LIC libraries exist for the genomes which have not been fully sequenced yet. In consequence, we expect that in the near future at least some parts of genome assembles will be done using clones whose nature lends itself to the analysis by this algorithm [3, 16], and to the extent they are present our approach would prove to be a valuable addition to any assembly quality assessment software toolkit.

Acknowledgments

The author would like to express gratitude to Ken Dewar, whose innovative ideas and diligence in leading the human genome closure effort at the Whitehead

Institute Center for Genome Research paved the way for this work, as well as to other members of the closure team who provided data and discussed relevant issues, especially Nathaniel Strauss, Christopher Seaman and Jean L. Chang. The early work on this algorithm has been partially supported by the Human Genome Project sequencing grant to Eric S. Lander and WICGR. We would also like to thank Rene Warren from the Genome Sciences Centre in Vancouver and David Messina from the Washington University School of Medicine for the data concerning the accuracy of recent assemblies and discussions during the Genome Informatics meeting at the Cold Spring Harbor Laboratory in November 2005.

References

1. Dunham, I., N. Shimizu, B. Roe *et al.* (1999) The DNA sequence of human chromosome 22. *Nature* **402**, 489–495.
2. Hattori, M., A. Fujiyama, T. Taylor *et al.* (2000) The DNA sequence of human chromosome 21. *Nature* **405**, 311–319.
3. Havlak, P., R. Chen, K.J. Durbin, A. Egan, Y. Ren, X.-Z. Song, G.M.Weinstock and R.A. Gibbs (2004) The Atlas genome assembly system. *Genome Res.* **14**, 721–732.
4. Huang, X., J. Wang, S. Aluru, S.-P. Yang and L. Hillier (2003) PCAP: A whole-genome assembly program. *Genome Res.* **13**, 2164–2170.
5. International Human Genome Mapping Consortium (2001) A physical map of the human genome. *Nature*, **409**, 934–941.
6. International Human Genome Sequencing Consortium (2001) Initial sequencing and analysis of the human genome. *Nature* **409**, 860–921.
7. International Human Genome Sequencing Consortium (2004) Finishing the euchromatic sequence of the human genome. *Nature*, **431**, 931–945.
8. Jaffe, D. B., J. Butler, S. Gnerre, E. Mauceli, K. Lindblad-Toh, J. P. Mesirov, M. C. Zody, and E. S. Lander (2003) Whole–genome sequence assembly for mammalian genomes: Arachne2. *Genome Res.* **13**, 91–96.
9. Kent, W. J. and D. Haussler (2001) Assembly of the working draft of the human genome with GigAssembler. *Genome Res.* **11**, 1541–1548.
10. Mouse Genome Sequencing Consortium (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature* **420**, 520–562.
11. Mullikin, J. and Z. Ning (2003). The Phusion assembler. *Genome Res.* **13**, 81–90.
12. Rat Genome Sequencing Consortium (2004) Genome sequence of the brown Norway rat yields insights into mammalian evolution. *Nature* **428**, 493–521.
13. Stojanovic, N., J. L. Chang, J. Lehoczky, M. C. Zody, and K. Dewar (2002) Identification of mixups among DNA sequencing plates. *Bioinformatics* **18**, 1418–1426.
14. Venter, J., M. Adams, E. Myers et al. (2001). The sequence of the human genome. *Science* **291**, 1304–1351.
15. Weber, J. L. and E. W. Myers (1997) Human whole–genome shotgun sequencing. *Genome Res.* **7**, 401–409.
16. Xu, J. and J.I. Gordon (2005) MapLinker: a software tool that aids physical map–linked whole genome shotgun assembly. *Bioinformatics* **21**, 1265–1266.

From HP Lattice Models to Real Proteins: Coordination Number Prediction Using Learning Classifier Systems

Michael Stout[1], Jaume Bacardit[1], Jonathan D. Hirst[2], Natalio Krasnogor[1], and Jacek Blazewicz[3]

[1] Automated Scheduling, Optimization and Planning research group, School of Computer Science and IT, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
{jqb, mqs, nxk}@cs.nott.ac.uk

[2] School of Chemistry, University of Nottingham, University Park, Nottingham NG7 2RD, UK
jonathan.hirst@nottingham.ac.uk

[3] Poznan University of Technology, Institute of Computing Science, ul. Piotrowo 3a, Poznan 60-965, Poland
jblazewicz@cs.put.poznan.pl

Abstract. Prediction of the coordination number (CN) of residues in proteins based solely on protein sequence has recently received renewed attention. At the same time, simplified protein models such as the HP model have been used to understand protein folding and protein structure prediction. These models represent the sequence of a protein using two residue types: hydrophobic and polar, and restrict the residue locations to those of a lattice. The aim of this paper is to compare CN prediction at three levels of abstraction a) 3D Cubic lattice HP model proteins, b) Real proteins represented by their HP sequence and c) Real proteins using residue sequence alone. For the 3D HP lattice model proteins the CN of each residue is simply the number of neighboring residues on the lattice. For the real proteins, we use a recent real-valued definition of CN proposed by Kinjo et al. To perform the predictions we use GAssist, a recent evolutionary computation based machine learning method belonging to the Learning Classifier System (LCS) family. Its performance was compared against some alternative learning techniques. Predictions using the HP sequence representation with only two residue types were only a little worse than those using a full 20 letter amino acid alphabet (64% vs 68% for two state prediction, 45% vs 50% for three state prediction and 30% vs 33% for five state prediction). That HP sequence information alone can result in predictions accuracies that are within 5% of those obtained using full residue type information indicates that hydrophobicity is a key determinant of CN and further justifies studies of simplified models.

1 Introduction

The prediction of the 3D structures of proteins is both a fundamental and difficult problem in computational biology. A popular approach to this problem is to predict some specific attributes of a protein, such as the secondary structure, the solvent accessibility or the coordination number. The coordination number (CN) problem is defined as the prediction, for a given residue, of the number of residues from the same protein that are in contact with it. Two residues are said to be in contact when the distance between the two is below a certain threshold. This problem is closely related to contact map (CM) prediction. It is generally believed that functional sites in proteins are formed from a pocket of residues termed an active site. Active site residues consist of a number of buried (high CN) residues hence studies of CN are of relevance to understanding protein function.

While protein structure prediction remains unsolved, researchers have resorted to simplified protein models to try to gain understanding of both the process of folding and the algorithms needed to predict it [1, 2, 3, 4, 5]. Approaches have included fuzzy sets, cellular automata, L-systems and memetic algorithms [6, 7, 8, 9, 10, 11]. One common simplification is to focus only on the residues (C-alpha or C-beta atoms) rather than all the atoms in the protein. A further simplification is to reduce the number of residue types to less than twenty by using residue sequence representations based, for instance, on physical properties such as hydrophobicity, as in the so called hydrophobic/polar (HP) models. Another simplification is to reduce the number of spatial degrees of freedom by restricting the atom or residue locations to those of a lattice [3, 5]. Lattices of various geometries have been explored, e.g., two-dimensional triangular and square geometries or three-dimensional diamond and face centered cubic [9].

The aim of this paper is to compare CN prediction for simplified HP lattice model proteins (Lattice-HP) with the prediction of the same feature for real proteins using either all twenty amino acid types (Real-AA) or using only the HP representation (Real-HP). This was done for several levels of class assignment (two state, three state and five state) and for a range of machine learning algorithms (LCS, C4.5 and NaiveBayes). The CN definition we use for real proteins was proposed recently by Kinjo et al.[12]. This is a continuous valued function, rather than the more frequently used discrete formulation [13].

The machine learning algorithm we focus on belongs to the family of Learning Classifier Systems (LCS) [14, 15], which are rule-based machine learning systems using evolutionary computation [16] as the search mechanism. Specifically, we have used a recent system called GAssist, which generates accurate, compact and highly interpretable solutions [17]. The performance of GAssist will be tested against some alternative learning mechanisms, and the performance of all these machine learning paradigms will be discussed.

2 Problem Definition

There is a large literature in CN/CM prediction, in which a variety of machine learning paradigms have been used, such as linear regression [12], neural networks [13], a combination of self-organizing maps and genetic programming [18] or support vector machines [19]. Several kinds of input information have been used in CN prediction besides the residue type of the residues in the chain, such as global information of the protein chain [12], data from multiple sequences alignments [13, 19, 18, 12] (mainly from PSI-BLAST [20]), predicted secondary structure [13, 19], predicted solvent accessibility [13] or sequence conservation [19].

There are also two main definitions of the distance used to determine whether there is contact between two residues. Some methods use the Euclidean distance between the C_α atoms of the two residues, while others use the C_β atom (C_α for glycine). Also, several methods discard the contacts between consecutive residues in the chain, and define a minimum chain separation as well as useing many different distance thresholds. Figure 1 shows a graphical representation of a non-local contact between two residues of a protein chain.

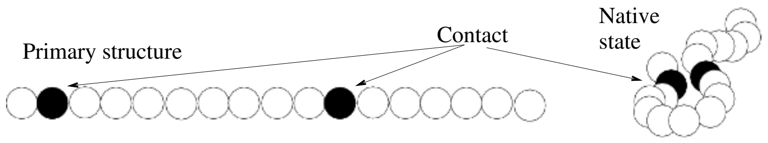


Fig. 1. Graphical representation of a non-local residue contact in a protein

Finally, there are two approaches to classification. Some methods predict the absolute CN, assigning a class to each possible value of CN. Other methods group instances [1] with close CN, for example, separating the instances with CNs lower or higher than the average of the training set, or defining classes in a way that guarantees uniform class distribution. We employ the latter approach as explained in section 2.3

2.1 HP Models

In the HP model (and its variants) the 20 residue types are reduced to two classes: non-polar or hydrophobic (H) and polar (P) or hydrophilic. An n residue protein is represented by a sequence $s \in \{H, P\}^+$ with $|s| = n$. The sequence s is mapped to a lattice, where each residue in s occupies a different lattice cell and the mapping is required to be self-avoiding. The energy potential in the HP model reflects the propensity of hydrophobic residues to form a hydrophobic core.

In the HP model, optimal (i.e. native) structures minimize the following energy potential:

$$E(s) = \sum_{i<j\ ;\ 1 \leq i,j \leq n} (\Delta_{i,j} \epsilon_{i,j}) \tag{1}$$

[1] For the rest of the paper the machine learning definition of instance is used: individual independent example of the concept to be learned [21]. That is, a set of features and the associated output (a class) that is to be predicted.

where

$$\Delta_{i,j} = \begin{cases} 1 \text{ if } i,j \text{ are in contact and } |i-j| > 1 \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

In the standard HP model, contacts that are HP and PP are assigned an energy of 0 and an HH contact is assigned an energy of -1.

2.2 Definition of CN

The distance used to determine contact by Kinjo et al. is defined using the C_β atom (C_α for glycine) of the residues. The boundary of the sphere defined by the distance cutoff $d_c \in \Re^+$ is made smooth by using a sigmoid function. Also, a minimum chain separation of two residues is required. Formally, the CN (O_i^p) of the residue i of protein chain p is computed as:

$$O_i^p = \sum_{j:|j-i|>2} \frac{1}{1 + exp(w(r_{ij} - d_c))} \tag{3}$$

where r_{ij} is the distance between the C_β atoms of the ith and jth residues. The constant w determines the sharpness of the boundary of the sphere. A value of three for w was used for all the experiments.

2.3 Conversion of the Real-Valued CN Definition into a Classification Domain

In order to convert the real-valued CN definition into a set of discrete states, so that it can be used as a classification dataset, Kinjo et al. propose a method to determine systematically some CN partitions resulting in an N class dataset. They choose the boundaries between classes in such a way as to generate classes with a uniform number of instances. They test two versions of this method. Defining the class boundaries separately for each residue type or defining them globally for all 20 residue types. In this study the later definition was adopted for simplicity and because it is more widely used.

3 The GAssist Learning Classifier System

GAssist [17] is a Pittsburgh Genetic–Based Machine Learning system descendant of GABIL [15]. The system applies a near-standard generational GA that evolves individuals that represent complete problem solutions. An individual consists of an ordered, variable–length rule set. A special fitness function based on the Minimum Description Length (MDL) principle [22] is used. The MDL principle is a metric applied in general to a theory (being a rule set here) which balances the complexity and accuracy of the rule set. The details and rationale of this fitness formula are explained in [17]. The system also uses a windowing scheme called ILAS (incremental learning with alternating strata) [23] to reduce the run-time of the system, especially for dataset with hundreds of thousands of instances as in this paper. We have used the GABIL [15] rule-based knowledge representation for nominal attributes and the adaptive discretization intervals (ADI) rule representation [17] for real-valued ones.

4 Experimental Framework

4.1 HP Lattice-Based Datasets

Two datasets were employed in this study, a 3D HP lattice model protein dataset and a data set of real proteins. Table 1 summarizes both datasets, which are available at `http://www.cs.nott.ac.uk/~nxk/hppdb.html`. For the Lattice-HP study, a set of structures from Hart's Tortilla Benchmark Collection (`http://www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html`) was used. This consisted of 15 structures on the simple cubic lattice (CN=6). Windows were generated for one, two and three residues at each side of a central residue and the CN class of the central residue assigned as the class of the instance. The instances was divided randomly into ten pairs of training and test sets These sets act in a similar way to a ten-fold cross-validation. The process was repeated ten times to create ten pairs of training and test sets. Each reported accuracy will be, therefore, the average of one hundred values.

Table 1. Details of the data sets used in these experiments

Name	Lattice-HP	K1050
Type	3D Cubic Lattice	Real Proteins
Number of Sequences	15	1050
Minimum Sequence Length	27	80
Maximum Sequence Length	48	2329
Total Hydrophobic	316	170493
Total Polar	309	84850
Total Residues	625	255343

4.2 Real Proteins Dataset

We have used the same dataset and training/test partitions used by Kinjo et al. [12]. The real protein dataset (Real-AA) was selected from PDB-REPRDB [24] with the following conditions: less than 30% sequence identity, sequence length greater than 50, no membrane proteins, no nonstandard residues, no chain breaks, resolution better than 2 Å and having a crystallographic R factor better than 20%. Chains that had no entry in the HSSP [25] database were discarded. The final data set contains 1050 protein chains. CN was computed using a distance cutoff of 10 Å. Windows were generated for one, two and three residues at each side of a central residue and the CN class of the central residue assigned as the class of the instance. The set was divided randomly into ten pairs of training and test set using 950 proteins for training and 100 for testing in each set. These sets act in a similar way to a ten-fold cross-validation. The proteins included in each partition are reported in `http://maccl01.genes.nig.ac.jp/~akinjo/sippre/suppl/list/`.

We have placed a copy of the dataset used in this paper at `http://www.asap.cs.nott.ac.uk/~jqb/EvoBIO_dataset.tar.gz(approx.85MB)`. This same dataset was used to generate a real protein HP sequence dataset (Real-HP) by assigning each residue a value of Hydrophobic or Polar as shown in Table 2, following Broome and Hecht [26].

Table 2. Assignment of residues as Hydrophobic or Polar

Residue (one letter code)	**Assignment**
ACFGILMPSTVWY	Hydrophobic
DEHKRQN	Polar

4.3 Attribute Distributions

For the Lattice-HP dataset, Figure 2 shows the distribution of hydrophobic/polar residues. Distributions are shown for a range of class assignments, two state, three state and five state. A higher proportion of hydrophobic residues are observed in the high CN classes, corresponding to a core of buried hydrophobic residues. A higher proportion of polar residues are found in the low CN (exposed) classes. This is not surprising, since these model protein structures have been optimized on the basis of hydrophobicity to group the hydrophobic residues together.

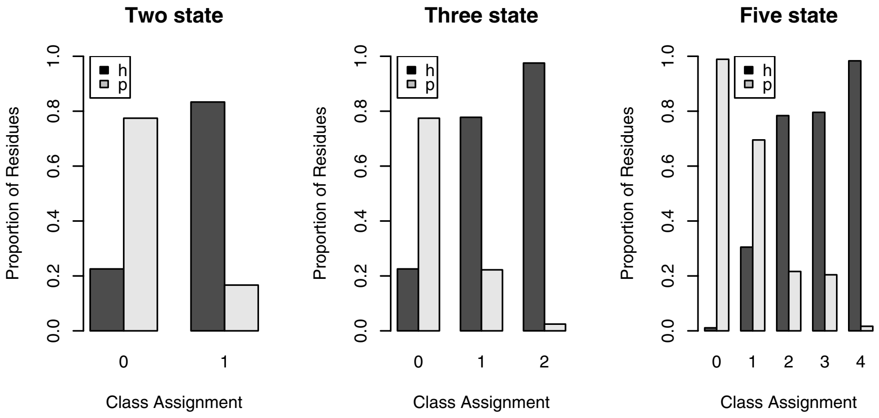


Fig. 2. Distribution of hydrophobic/polar residues in the Lattice-HP dataset: h=hydrophobic, p=polar

For the Real-HP dataset, Figure 3 shows the distribution of hydrophobic/polar residues two state, three state and five state class assignments. In these distributions hydrophobic residues are significantly more prevalent in the high CN classes, corresponding to a core of buried hydrophobic residues. The approximately equal distribution of hydrophobic and polar residues observed in the low CN classes (corresponding to exposed/surface residues) may stem from the approximately two hydrophobic to one polar assignment ratio in Table 2. These distributions provide a baseline against which the performance of the prediction algorithms can be gauged.

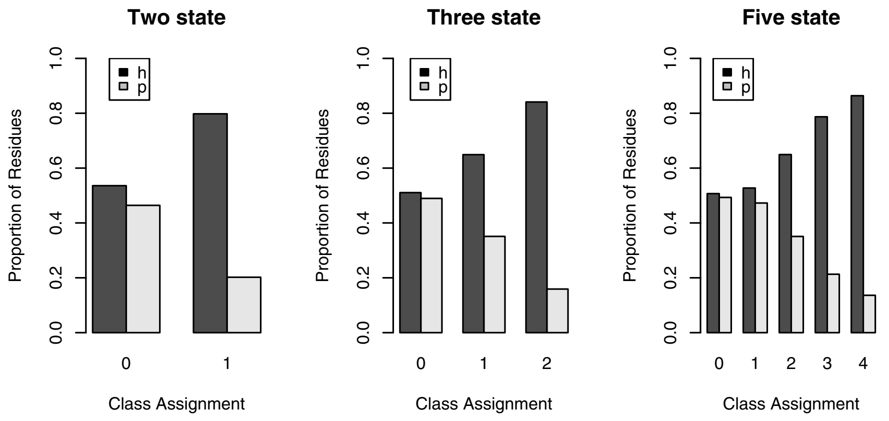


Fig. 3. Distribution of hydrophobic/polar residues in the Real-HP dataset: h=hydrophobic, p=polar

5 Results

The performance of GAssist was compared to two other machine learning systems: C4.5 [27], a rule induction system and Naive Bayes [28], a Bayesian learning algorithm. The WEKA [21] implementation of these algorithms was used. Student t-tests were applied to the mean prediction accuracies (rather than individual experimental data points) to determine, for each dataset, those algorithms that significantly outperformed other methods using a confidence interval of 95% and Bonferroni correction [29] for multiple pair-wise comparisons was used.

5.1 Lattice-HP Datasets

Table 3 compares the results of two, three and five state CN predictions for a range of window sizes for the GAssist LCS, Naive Bayes and C4.5 using the Lattice-HP dataset. A window size of three means three residues either side of the central residue, i.e. a seven residue peptide. As the number of states is increased the accuracy decreases from around 80% to around 51% for all algorithms. For each state as the window size is increased the accuracy increases by around 0.1-0.2%. With the exception of the C4.5 algorithm which shows a decrease in accuracy with increasing window size in two and three state predictions. There were no significant differences detected in these tests.

For two states, the best prediction was given by C4.5 with window size of one (80%±4.9). For three states the best prediction was given by GAssist with window size of two (67%±4.1). For five states GAssist again gave the best predictions for a window size of three (52.7%±5.3).

Table 3. Lattice-HP Prediction Accuracies

Number of States	Algorithm	Window Size		
		1	2	3
2	GAssist	79.8 ±4.9	80.2 ±5.0	80.0 ±5.3
	C4.5	80.2 ±4.9	79.9 ±5.0	79.7 ±5.1
	NaiveBayes	79.8 ±4.9	80.0 ±4.9	80.2 ±5.0
3	GAssist	67.4 ±4.9	67.8 ±4.1	67.3 ±5.0
	C4.5	67.5 ±4.8	67.6 ±4.2	66.6 ±5.0
	NaiveBayes	67.2 ±4.6	67.3 ±4.4	67.5 ±4.8
5	GAssist	51.4 ±4.6	51.3 ±4.2	52.7 ±5.3
	C4.5	51.7 ±4.5	51.0 ±4.1	52.2 ±5.1
	NaiveBayes	51.7 ±4.6	52.3 ±4.3	51.9 ±5.6

5.2 Real Proteins

Table 4 compares the results of two, three and five state CN predictions on real proteins for the GAssist LCS, Naive Bayes and C4.5 for the Real-HP dataset. When an HP sequence representation was used, an increase in the number of states is accompanied by a decrease in accuracy from around 63-64% to around 29-30% for all algorithms. For each state, as the window size is increased the accuracy increases by around 1%. For two states, the best predictions were given by GAssist and C4.5 with window size of three (64.4%±0.5). For three states the best prediction was given by C4.5 with window size of two (45%±0.4). For five states C4.5 again gave the best predictions for a window size of three (30.4%±0.5).

Table 4. CN Prediction Accuracies for the Real-HP and Real-AA datasets. A • means that GAssist outperformed the Algorithm to the left (5% t-test significance). A ∘ label means that the Algorithm on the left outperformed GAssist (5% t-test significance).

State	Algorithm	HP Based			Residue Based		
		Window Size			Window Size		
		1	2	3	1	2	3
2	GAssist	63.6±0.6	63.9±0.6	64.4±0.5	67.5±0.4	67.9±0.4	68.2±0.4
	C4.5	63.6±0.6	63.9±0.6	64.4±0.5	67.3±0.4	67.5±0.3	67.8±0.3
	NaiveBayes	63.6±0.6	63.9±0.6	64.3±0.5	67.6±0.4	68.0±0.4	68.8±0.3∘
3	GAssist	44.9±0.5	45.1±0.5	45.6±0.4	48.8±0.4	49.0±0.4	49.3±0.4
	C4.5	44.9±0.5	45.1±0.5	45.8±0.4	48.8±0.3	48.7±0.3	49.1±0.3
	NaiveBayes	44.7±0.5	45.2±0.5	45.7±0.4	49.0±0.4	49.6±0.5∘	50.7±0.3∘
5	GAssist	29.0±0.3	29.6±0.5	30.1±0.5	32.2±0.3	32.5±0.3	32.7±0.4
	C4.5	29.0±0.3	29.7±0.4	30.4±0.5	31.9±0.4	31.4±0.4•	31.0±0.5•
	NaiveBayes	29.0±0.3	29.7±0.4	30.1±0.5	33.0±0.2∘	33.9±0.3∘	34.7±0.4∘

Using full residue information, an increase in the number of states is accompanied by a decrease in accuracy from around 68% to around 34% for all algorithms. For each state, as the window size is increased, the accuracy increases by around 0.5%, with the exception of the C4.5 algorithm which shows a decrease in accuracy with increasing window size in five state predictions. The LCS outperformed C4.5 two times and was outperformed by Naive Bayes six times. For two, three and five state predictions the best results were given by Naive Bayes with window size of three (68.8%±0.3, 50.7%±0.3 and 34.7%±0.4 respectively). Most interestingly, moving from HP sequence representation to full residue type sequence information only results in a 4% increase for two and three state and 1-2% increase for, the more informative, five state prediction.

5.3 Brief Estimation of Information Loss

In order to understand the effect of using a lower-dimensionality profile of a protein chain such as the HP model, we have computed some simple statistics on the datasets. Two measures are computed:

$$\text{redundancy} = 1 - \frac{\#\text{unique instances}}{\#\text{total instances}} \tag{4}$$

$$\text{inconsistency} = \frac{\left(\frac{\#\text{unique instances}}{\#\text{unique antecedents}}\right) - 1}{\#states - 1} \tag{5}$$

Equation 4 shows the effect of reducing the alphabet and the window size: creating many copies of the same instances. Equation 5 shows how this reduction creates inconsistent instances: instances with equal input attributes (antecedent) but different class. For the sake of clarity this measure has been normalized for the different number of target states. Table 5 shows these ratios. For two-states and window size of one, the Real-HP dataset shows the most extreme case: any possible antecedent appears in the data set associated to both classes. Fortunately, the proportions of the two classes for each antecedent are different, and the system can still learn. We see how the Real-HP dataset is highly redundant and how the Real-AA dataset of window size two and three presents low redundancy and inconsistency rate.

Table 5. Redundancy and inconsistency rate of the tested real-proteins datasets

		HP representation		AA representation	
States	Window Size	Redundancy	Inconsistency	Redundancy	Inconsistency
2	1	99.99%	100.000%	93.69%	90.02%
	2	99.94%	92.50%	6.14%	3.85%
	3	99.75%	81.71%	0.21%	0.05%
3	1	99.98%	96.88%	90.90%	87.01%
	2	99.92%	86.25%	4.50%	2.84%
	3	99.66%	76.00%	0.17%	0.04%
5	1	99.97%	93.75%	85.84%	81.52%
	2	99.86%	86.25%	2.97%	1.84%
	3	99.46%	74.36%	0.14%	0.03%

6 Discussion

The LCS and other machine learning algorithms preformed at similar levels for these CN prediction tasks. Generally, increasing the number of classes (number of states) leads to a reduction in prediction accuracy which can be partly offset by using a larger window size. Reduction of input information from full residue type to HP sequence reduces the accuracy of prediction. The algorithms were, however, all capable of predictions using HP sequence that were within 5% of the accuracies obtained using full residue type sequences.

For all of the algorithms studied, in the case of the most informative five state predictions, moving from HP lattice to real protein HP sequences leads to a reduction of CN prediction accuracy from levels of around 50% to levels of around 30%. The significant reduction in the spatial degrees of freedom in the Lattice-HP models leads to an improvement in prediction accuracy of around 20%.

In contrast, moving from the real protein HP sequences to real protein full residue type sequences (for the same five state CN predictions) only a 3-5% improvement in prediction accuracy results from inclusion of this additional residue type information. This seems to indicate that hydrophobicity information is a key determinant of CN and that algorithmic studies of HP models are relevant. The rules that result from a reduced two letter alphabet are simpler and easier to understand than those from the full residue type studies. For example, for the HP representation a rule set giving 62.9% accuracy is shown below (an X symbol is used to represent positions at the end of the chains, that is beyond the central residue being studied).

1. If $AA_{-1} \notin \{x\}$ and $AA \in \{h\}$ and $AA_1 \in \{p\}$ then class is 1
2. If $AA_{-1} \in \{h\}$ and $AA \in \{h\}$ and $AA_1 \notin \{x\}$ then class is 1
3. If $AA_{-1} \in \{p\}$ and $AA \in \{h\}$ and $AA_1 \in \{h\}$ then class is 1
4. Default class is 0

In these rules, a class assignment of high is represented by 1 and low by 0. For the full residue type representation a rule set giving 67.7% accuracy is:

1. If $AA_{-1} \notin \{D, E, K, N, P, Q, R, S, X\}$ and $AA \notin \{D, E, K, N, P, Q, R, S, T\}$ and $AA_1 \notin \{D, E, K, Q, X\}$ then class is 1
2. If $AA_{-1} \notin \{X\}$ and $AA \in \{A, C, F, I, L, M, V, W, Y\}$ and $AA_1 \notin \{D, E, H, Q, S, X\}$ then class is 1
3. If $AA_{-1} \notin \{P, X, Y\}$ and $AA \in \{A, C, F, I, L, M, V, W, Y\}$ and $AA_1 \notin \{K, M, T, W, X, Y\}$ then class is 1
4. If $AA_{-1} \notin \{H, I, K, M, X\}$ and $AA \in \{C, F, I, L, M, V, W, Y\}$ and $AA_1 \notin \{M, X\}$ then class is 1
5. Default class is 0

Recently, Kinjo et al [12] reported two, three and ten state CN prediction at accuracies of 72.1%, 53.7%, and 18.8% respectively, which is higher than our results. However, they use a non-standard accuracy measure that usually gives slightly higher results than the one used in this paper. Also, they use more input information than was used in the experiments reported in this paper.

The aim of this paper was to compare the performance difference between the Real-AA and Real-HP representations, not to obtain the best CN results. We have undertaken more detailed studies on both the HP model dataset for CN and Residue Burial prediction and the real protein datasets for CN prediction in comparison to the Kinjo work (papers submitted).

7 Conclusions and Further Work

This paper has shown that it is possible to predict residue CN for HP Lattice model proteins at a level of around 52% for five state prediction using a window of three residues either side of the prediced residue. For real proteins, five state CN prediction using a window size of three can be performed at a level of 30% using HP residue profiles. This can be increased to 32% using full sequence information. This is perhaps understandable since reducing the sequence to an HP sequence discards useful information. However, the representation with only two residue types is only a little worse than that with a full twenty letter alphabet (64% vs 68% for two state prediction, 45% vs 50% for three state prediction and 30% vs 33% for five state prediction). Thus, most of the information is contained in the HP representation, indicating that hydrophobicity is a key determinant of CN. This is consistent with earlier studies [30].

Initial estimates of information inconsistency (ambiguous antecedent to consequent assignments) in the reduced two letter alphabet dataset indicate that considerable inconsistency is present even for five state assignments using larger window sizes. The algorithms presumably learn from the various distributions of these inconsistencies during their learning stage. Li et al. [31] have investigated whether there is a minimal residue type alphabet by which proteins can be folded. They conclude that a ten letter alphabet may be sufficient to characterize the complexity of proteins. We are performing studies to investigate such reduced letter alphabets and to quantify the information loss in each. In future, we will extend these studies to prediction of other structural attributes, such as secondary structure and relative solvent accessibility. These studies will help determine the relative utility of CN for designing prediction heuristics for HP models and Real proteins.

Acknowledgments

We acknowledge the support provided by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant GR/T07534/01 and the Biotechnology and Biological Sciences Research Council (BBSRC) under grant BB/C511764/1.

References

1. Abe, H., Go, N.: Noninteracting local-structure model of folding and unfolding transition in globular proteins. ii. application to two-dimensional lattice proteins. Biopolymers **20** (1981) 1013–1031
2. Hart, W.E., Istrail, S.: Crystallographical universal approximability: A complexity theory of protein folding algorithms on crystal lattices. Technical Report SAND95-1294, Sandia National Labs, Albuquerque, NM (1995)
3. Hinds, D., Levitt, M.: A lattice model for protein structure prediction at low resolution. In: Proceedings National Academy of Science U.S.A. Volume 89. (1992) 2536–2540

4. Hart, W., Istrail, S.: Robust proofs of NP-hardness for protein folding: General lattices and energy potentials. Journal of Computational Biology (1997) 1–20
5. Yue, K., Fiebig, K.M., Thomas, P.D., Sun, C.H., Shakhnovich, E.I., Dill, K.A.: A test of lattice protein folding algorithms. Proc. Natl. Acad. Sci. USA **92** (1995) 325–329
6. Escuela, G., Ochoa, G., Krasnogor, N.: Evolving l-systems to capture protein structure native conformations. In: Proceedings of the 8th European Conference on Genetic Programming (EuroGP 2005), Lecture Notes in Computer Sciences 3447, pp 73-84, Springer-Verlag, Berlin (2005)
7. Krasnogor, N., Pelta, D.: Fuzzy memes in multimeme algorithms: a fuzzy-evolutionary hybrid. In Verdegay, J., ed.: Fuzzy Sets based Heuristics for Optimization, Springer (2002)
8. Krasnogor, N., Hart, W., Smith, J., Pelta, D.: Protein structure prediction with evolutionary algorithms. In Banzhaf, W., Daida, J., Eiben, A., Garzon, M., Honavar, V., Jakaiela, M., Smith, R., eds.: GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (1999)
9. Krasnogor, N., Blackburne, B., Burke, E., Hirst, J.: Multimeme algorithms for protein structure prediction. In: Proceedings of the Parallel Problem Solving from Nature VII. Lecture Notes in Computer Science. Volume 2439. (2002) 769–778
10. Krasnogor, N., de la Cananl, E., Pelta, D., Marcos, D., Risi, W.: Encoding and crossover mismatch in a molecular design problem. In Bentley, P., ed.: AID98: Proceedings of the Workshop on Artificial Intelligence in Design 1998. (1998)
11. Krasnogor, N., Pelta, D., Marcos, D.H., Risi, W.A.: Protein structure prediction as a complex adaptive system. In: Proceedings of Frontiers in Evolutionary Algorithms 1998. (1998)
12. Kinjo, A.R., Horimoto, K., Nishikawa, K.: Predicting absolute contact numbers of native protein structure from amino acid sequence. Proteins **58** (2005) 158–165
13. Baldi, P., Pollastri, G.: The principled design of large-scale recursive neural network architectures dag-rnns and the protein structure prediction problem. Journal of Machine Learning Research **4** (2003) 575 – 602
14. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3** (1995) 149–175
15. DeJong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithms for concept learning. Machine Learning **13** (1993) 161–188
16. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
17. Bacardit, J.: Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time. PhD thesis, Ramon Llull University, Barcelona, Catalonia, Spain (2004)
18. MacCallum, R.: Striped sheets and protein contact prediction. Bioinformatics **20** (2004) I224–I231
19. Zhao, Y., Karypis, G.: Prediction of contact maps using support vector machines. In: Proceedings of the IEEE Symposium on BioInformatics and BioEngineering, IEEE Computer Society (2003) 26–36
20. Altschul, S.F., Madden, T.L., Scher, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic Acids Res **25** (1997) 3389–3402
21. Witten, I.H., Frank, E.: Data Mining: practical machine learning tools and techniques with java implementations. Morgan Kaufmann (2000)
22. Rissanen, J.: Modeling by shortest data description. Automatica **vol. 14** (1978) 465–471

23. Bacardit, J., Goldberg, D., Butz, M., Llorà, X., Garrell, J.M.: Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy. In: Parallel Problem Solving from Nature - PPSN 2004, Springer-Verlag, LNCS 3242 (2004) 1021–1031
24. Noguchi, T., Matsuda, H., Akiyama, Y.: Pdb-reprdb: a database of representative protein chains from the protein data bank (pdb). Nucleic Acids Res **29** (2001) 219–220
25. Sander, C., Schneider, R.: Database of homology-derived protein structures. Proteins **9** (1991) 56–68
26. Broome, B., Hecht, M.: Nature disfavors sequences of alternating polar and non-polar amino acids: implications for amyloidogenesis. J Mol Biol **296** (2000) 961–968
27. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
28. John, G.H., Langley, P.: Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo (1995) 338–345
29. Miller, R.G.: Simultaneous Statistical Inference. Springer Verlag, New York (1981) Heidelberger, Berlin.
30. Miller, S., Janin, J., Lesk, A., Chothia, C.: Interior and surface of monomeric proteins. J Mol Biol **196** (1987) 641–656
31. Li, T., Fan, K., Wang, J., Wang, W.: Reduction of protein sequence complexity by residue grouping. Protein Eng **16** (2003) 323–330

Conditional Random Fields for Predicting and Analyzing Histone Occupancy, Acetylation and Methylation Areas in DNA Sequences

Dang Hung Tran[1], Tho Hoan Pham[2], Kenji Satou[1,3], and Tu Bao Ho[1,3]

[1] School of Knowledge Science, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
tran@jaist.ac.jp
[2] Faculty of Information Technology, Hanoi University of Pedagogy, 136 Xuan Thuy, Cau Giay, Hanoi, Vietnam
[3] Institute for Bioinformatics Research and Development (BIRD), Japan Science and Technology Agency (JST), Japan

Abstract. Eukaryotic genomes are packaged by the wrapping of DNA around histone octamers to form nucleosomes. Nucleosome occupancies together with their acetylation and methylation are important modification factors on all nuclear processes involving DNA. There have been recently many studies of mapping these modifications in DNA sequences and of relationship between them and various genetic activities, such as transcription, DNA repair, and DNA remodeling. However, most of these studies are experimental approaches. In this paper, we introduce a computational approach to both predicting and analyzing nucleosome occupancy, acetylation, and methylation areas in DNA sequences. Our method employs conditional random fields (CRFs) to discriminate between DNA areas with high and low relative occupancy, acetylation, or methylation; and rank features of DNA sequences based on their weight in the CRFs model trained from the datasets of these DNA modifications. The results from our method on the yeast genome reveal genetic area preferences of nucleosome occupancy, acetylation, and methylation are consistent with previous studies.

Keywords: Histone proteins, acetylation, methylation, conditional random fields.

1 Introduction

Eukaryotic genomes are packaged into nucleosomes that consist of 145–147 base pairs of DNA wrapped around a histone octamer [9]. The histone components of nucleosomes and their modification state (of which acetylation and methylation are the most important ones) can profoundly influence many genetic activities, including transcription [2, 4, 5, 16], DNA repair, and DNA remodeling [13].

There have been recently many studies of mapping histone occupancies together with their modifications in DNA sequences and of relationship between them and various genetic activities concerning DNAs [1, 2, 5, 7, 16, 18, 19]. But most of these studies were experimentally conducted by the combination of chromatin immunoprecipitation and whole-genome DNA microarrays, or ChIP-Chip protocol.

The nucleosome occupancy as well as its modifications such as acetylation and methylation mainly depend on the DNA sequence area they incorporate in. The majority of acetylation and methylation occurs at specific highly conserved residues in the histone components of nucleosomes: acetylation sites include at least nine lysines in histone H3 and H4 (H3K9, H3K14, H3K18, H3K23, H3K27, H4K5, H4K8, H4K12, and H4K16); methylation sites include H3K4, H3K9, H3K27, H3K36, H3K79, H3R17, H4K20, H4K59, H4R3 [14]. When a nucleosome appears in a specific DNA sequence area, these potentially sites can have a certain acetylation or methylation level [5, 16].

Recently we have introduced a support vector machine (SVM)-based method to qualitatively predict histone occupancy, acetylation and methylation areas in DNA sequences [15]. In this paper, we present a different computational method for this prediction problem. We employ conditional random fields (CRF) [6], a novel machine learning technique, to discriminate between DNA areas with high and low relative occupancy, acetylation, or methylation. Our experiments showed that CRF-based method has competitive performance with SVM method. Moreover, similar to SVMs, our CRF method can extract informative k-gram features based on their weight in the CRFs model trained from the datasets of these DNA modifications. The results from our CRF-method on the yeast genome are consistent with those from the SVM method and reveal genetic area preferences of nucleosome occupancy, acetylation, and methylation that are consistent with previous studies.

2 Materials and Methods

2.1 Datasets

From the genome-wide map of nucleosome acetylation and methylation reported in [16], we extracted 14 datasets and used to illustrate the performance of our method. These datasets are described in detail in Table 1. Each example in the datasets corresponds to a DNA sequence area (segment) with a fixed length L (in our experiments, we selected $L = 200, 500, 1000, 1500$). A DNA sequence area is assigned to the positive class if the relative occupancy, acetylation, or methylation [16] measured at its middle position is greater than 1.2, and to the negative class if the relative occupancy, acetylation, or methylation is lesser than 0.8. Sequences with value in between 0.8 and 1.2 are ignored.

2.2 Conditional Random Fields

The sequential classification problem is well known in several scientific fields, especially computational linguistics, and computational biology [6]. There are

Table 1. Datasets of histone occupancy, acetylation, and methylation by ChIP-Chip protocol in vivo [16]

Dataset	#positives	#negatives	Description
H3.YPD	7667	7298	H3 occupancy
H4.YPD	6480	8121	H4 occupancy
H3.H2O2	17971	15516	H3.H202 occupancy
H3K9acvsH3.YPD	15415	12367	H3K9 acetylation relative to H3
H3K14acvsH3.YPD	18771	14277	H3K14 acetylation relative to H3
H3K14acvsWCE.YPD	17672	16290	H3K14 acetylation relative to WCE
H3K14acvsH3.H2O2	18410	15685	H3K14 acetylation relative to H3.H2O2
H4acvsH3.YPD	18410	15685	H4 acetylation relative to H3
H4acvsH3.H2O2	18143	12540	H4 acetylation relative to H3.H2O2
H3K4me1vsH3.YPD	17266	14411	H3K4 monomethylation relative to H3
H3K4me2vsH3.YPD	18143	12540	H3K4 dimethylation relative to H3
H3K4me3vsH3.YPD	19604	17195	H3K4 trimethylation relative to H3
H3K36me3vsH3.YPD	18892	15988	H3K36 trimethylation relative to H3
H3K79me3vsH3.YPD	15337	13500	H3K79 trimethylation relative to H3

two kinds of model for solving this problem, generative models and conditional models. While generative models define a joint probability distribution of the observation and labelling sequences $p(X, Y)$, the conditional models specify the probability of a label given an observation sequence $p(Y|X)$. The main drawback in generative models is that, in order to define a joint probability distribution, they must enumerate all possible observation sequences, which may be not feasible in practice [6, 12, 21]. Our work employs conditional models, specially conditional random fields, which can overcome the drawbacks of generative models.

CRF [6] is a probabilistic framework for segmenting and labelling sequential data using conditional model [6]. It has the form of a undirected graph that defines a log-linear distribution over label sequences given a particular observation sequence. CRFs have several advantages over other models (e.g., HMMs and MEMMs) such as relaxing strong independence Markov assumptions and avoiding weakness called the label bias problem [6, 11, 12, 21].

Definition. CRFs can be represented by an undirected graphical model. According to [6], we define $G = (V, E)$ to be an undirected graph, with $v \in V$ corresponds to each of the random variables representing a label sequence Y_v from Y, and $e \in E$ corresponds to the definition of conditional independence for undirected graphical models. In other words, two vertices v_i and v_j are conditionally independent given all other random variables in the graph.

In theory, CRFs can be represented by arbitrarily structure graph, although in this work, we focus on linear-chain structure graph. Let $X = (x_1, x_2, ..., x_T)$ be an observed data sequence; S be a set of finite state machines, each is associated with a label $l \in L$; and $Y = (y_1, y_2, ..., y_T)$ be the state sequence. The linear-chain CRFs [20, 12] then define the conditional probability of a state sequence given an input sequence as follows

$$p_\theta(Y|X) = \frac{1}{Z(X)} exp(\sum_{i=1}^{T} \sum_k \lambda_k f_k(y_{i-1}, y_i, X, i))$$

where $Z(X) = \sum_{s \in S} exp(\sum_{i=1}^{T} \sum_k \lambda_k f_k(y_{i-1}, y_i, X, i))$ is a normalization factor over all state sequences, and $f_k(y_{i-1}, y_i, X, i)$ are feature functions, each of them is either a state feature function or a transition function [20, 12, 21]. A state feature captures a particular property of the observation sequence X at current state y_i. A transition feature represents sequential dependencies by combining the label l' of the previous state y_{i-1} and the label l of the current state y_i. As [6], we assume that the feature functions is fixed, and denote $\lambda = \{\lambda_k\}$ as a weight vector which to be learned through training.

Inference in CRFs. Inference in CRFs is to find a state sequence y^* which is the most likely given the observation sequence x

$$y^* = \text{argmax}_y p_\theta(y|x) = \text{argmax}_y \left\{ exp(\sum_{i=1}^{T} \sum_k \lambda_k f_k(y_{i-1}, y_i, x, i)) \right\}$$

Similarly to HMMs, CRFs use a dynamic programming method for finding y^* [6, 21, 12]. In fact, we choose the most well-known method being the Vieterbi algorithm [17]. Viterbi stores the probability of the most likely path up to time t which accounts for the first t observations and ends in state y_t. We define this probability to be $\alpha_t(y_i)$ $(0 \leq t \leq T-1)$. We set $\alpha_0(y_i)$ to be the probability of starting in state y_i. The recursion is given by

$$\alpha_{t+1} = max_{y_j} \left\{ \alpha_t(y_j) exp\left(\sum_k \lambda_k f_k(y_j, y_i, x, t)\right) \right\}$$

At the end time (i.e., $t = T-1$), we can backtrack through the stored information to find the most likely sequence y^*.

Training CRFs. Let $D = \left\{(x^k, y^k)\right\}_{k=1}^{N}$ be the training data set. CRFs are trained by finding the weight vector $\theta = \{\lambda_1, \lambda_2, ...\}$ to maximize the log-likelihood

$$L = \sum_{j=1}^{N} log\left(p_\theta(y^{(j)}|x^{(j)})\right) - \sum_k \frac{\lambda^2}{2\sigma^2}$$

where the second sum is a Gaussian prior over parameters (with variance σ^2) that provides smoothing to help coping with sparsity in the training data [3].

Since the likelihood function in exponential models of CRFs is convex, the above optimization problem always has the global optimum solution, which can be found by an iterated estimation procedure. The traditional method for training in CRFs is iterative scaling algorithms [6, 21]. Sine those methods are very slow for classification [20], therefore we use quasi-Newton methods, such as L-BFGS [8], which are significantly more efficient [10, 20].

L-BFGS is a limited-memory quasi-Newton procedure for unconstrained optimization that requires the value and gradient vector of a function to be optimized. Assuming that the training labels on instance j make its state path unambiguous, let $y^{(j)}$ denote that path, then the first-derivative of the log-likelihood is

$$\frac{\delta L}{\delta \lambda_k} = \left(\sum_{j=1}^{N} C_k(y^{(j)}, x^{(j)})\right) - \left(\sum_{j=1}^{N} \sum_y p_\theta(y|x^{(j)}) C_k(y, x^{(j)})\right) - \frac{\lambda_k}{\sigma^2}$$

where $C_k(y, x)$, the count of feature f_k given y and x, equal to $\sum_{t=1}^{T} f_k(y_{i-1}, y_i, x, i)$, i.e., the sum of $f_k(y_{i-1}, y_i, x, i)$ values for all positions i in the training sequence. The first two terms correspond to the difference between the empirical and the model expected values of feature f_k. The last term is the first-derivative of the Gaussian prior.

2.3 Features of a DNA Sequence Area

The most important issue in CRFs learning is to select a set of features that hopefully capture the relevant relationships among observations and label sequences. CRFs have two kinds of features, state features and transition features. However, in this work we focus only on state features. Also, each observation sequence in the datasets has only one observation (L-DNA sequence area) and the label sequence is a sequence of 0 (negative class) and 1 (positive class). Our feature set to input to CRF systems is built by two steps. First, we use a k-sliding window along a DNA sequence to get binary k-grams (patterns of k consecutive nucleotide symbols). Each DNA sequence is thus represented by a binary 4^k-dimensional vector of all possible k-grams. Second, we define the unigram function for each k-gram as follows:

$$u_t(x) = \begin{cases} 1 \text{ if the t}^{th}\ k\text{-gram appear in the sequence } x \\ \\ 0 \text{ otherwise} \end{cases}$$

Therefore, the relationship between the observation and two classes, positive and negative, is described in the following features:

$$f_{tP}(y, x) = \begin{cases} u_t(x) \text{ if } y \text{ belong to positive class} \\ \\ 0 \text{ otherwise} \end{cases}$$

$$f_{tN}(y, x) = \begin{cases} u_t(x) \text{ if } y \text{ belong to negative class} \\ \\ 0 \text{ otherwise} \end{cases}$$

3 Results and Discussion

3.1 Prediction of Histone Occupancy, Acetylation, and Methylation

We used CRFs with the limited-memory quasi-Newton method (Section 2.2) to perform threefold cross-validation on 14 datasets of histone occupancy, acetylation and methylation areas (Table 1). Three criteria of precision, recall and F1-measure are used to report the results:

Table 2. Results of histone occupancy, acetylation and methylation prediction

	k=5			*k*=6			*k*=4,5			*k*=5,6		
Dataset	*Pre.*	*Rec.*	*F*1.	*Pre.*	*Rec.*	*F*1.	*Pre.*	*Rec.*	*F*1.	*Pre.*	*Rec.*	*F*1.
H3.YPD	80.17	80.17	80.07	82.33	82.31	82.32	80.27	80.27	80.27	**82.67**	**82.46**	**82.56**
	(80.54)	(80.50)	(80.52)	(81.78)	(81.62)	(81.70)	(80.94)	(80.89)	(80.92)	(79.97)	(79.28)	(79.62)
H4.YPD	83.21	83.07	83.14	85.62	85.49	85.55	82.79	82.67	82.73	**85.57**	**85.53**	**85.55**
	(81.72)	(81.57)	(81.65)	(83.97)	(83.87)	(83.92)	(81.87)	(81.72)	(81.79)	(82.63)	(82.13)	(82.38)
H3.H2O2	82.80	82.53	82.67	82.96	82.98	82.97	82.73	82.76	82.74	**83.06**	**83.00**	**83.03**
	(80.85)	(80.79)	(80.82)	(81.51)	(81.34)	(81.43)	(81.20)	(81.12)	(81.16)	(81.45)	(81.39)	(81.42)
H3K9acvsH3.YPD	70.36	70.22	70.29	71.50	71.27	71.38	69.86	69.65	69.75	**71.58**	**71.44**	**71.51**
	(70.92)	(70.74)	(70.83)	(73.98)	(73.49)	(73.74)	(71.12)	(70.96)	(71.04)	(71.76)	(70.22)	(70.98)
H3K14acvsH3.YPD	66.58	65.99	66.28	**68.13**	**68.06**	**68.09**	66.26	65.69	65.97	68.27	67.78	68.02
	(68.55)	(67.66)	(68.10)	(73.12)	(71.68)	(72.39)	(68.80)	(67.80)	(68.30)	(71.95)	(67.44)	(69.62)
H3K14acvsWCE.YPD	62.86	62.60	62.73	63.76	63.67	63.71	65.88	65.69	65.79	**63.74**	**63.72**	**63.73**
	(64.04)	(63.95)	(64.00)	(68.04)	(67.83)	(67.94)	(64.47)	(64.37)	(64.42)	(66.52)	(65.81)	(66.16)
H3K14acvsH3.H2O2	65.42	65.41	65.41	**66.58**	**66.44**	**66.51**	66.21	66.02	66.12	66.46	66.43	66.44
	(67.14)	(67.13)	(67.14)	(69.88)	(69.44)	(69.66)	(67.35)	(67.34)	(67.34)	(69.28)	(68.85)	(69.06)
H4acvsH3.YPD	66.21	66.02	66.12	67.64	67.43	67.53	65.94	65.75	65.85	**67.85**	**67.66**	**67.75**
	(68.01)	(67.62)	(67.82)	(72.22)	(71.60)	(71.91)	(68.31)	(67.85)	(68.08)	(69.98)	(68.11)	(69.03)
H4acvsH3.H2O2	69.73	69.69	69.71	**70.44**	**70.27**	**70.35**	69.65	69.59	69.62	70.16	70.11	70.13
	(69.32)	(69.27)	(69.29)	(71.69)	(71.42)	(71.55)	(69.32)	(69.25)	(69.29)	(69.93)	(69.07)	(69.50)
H3K4me1vsH3.YPD	65.21	64.79	65.00	66.47	65.83	66.15	64.87	64.40	64.63	**67.03**	**66.34**	**66.68**
	(66.16)	(65.59)	(65.87)	(69.55)	(68.48)	(69.01)	(66.43)	(65.79)	(66.11)	(68.30)	(65.87)	(67.06)
H3K4me2vsH3.YPD	68.58	68.05	68.31	**64.78**	**62.85**	**63.80**	62.95	62.15	62.54	64.41	62.95	63.67
	(64.20)	(61.50)	(62.82)	(68.10)	(64.15)	(66.07)	(64.68)	(61.70)	(63.16)	(67.25)	(60.18)	(63.52)
H3K4me3vsH3.YPD	67.00	66.81	66.91	63.18	62.66	62.92	60.90	60.53	60.71	**(64.49)**	**64.23**	**64.36**
	(63.96)	(63.55)	(63.75)	(68.90)	(64.32)	(66.53)	(64.50)	(64.06)	(64.28)	(68.08)	(65.76)	(66.90)
H3K36me3vsH3.YPD	69.76	69.55	69.65	71.21	71.05	71.13	69.46	69.24	69.35	**71.44**	**70.90**	**71.17**
	(70.85)	(70.61)	(70.73)	(72.39)	(71.93)	(72.16)	(71.11)	(70.87)	(70.99)	(72.83)	(71.88)	(72.36)
H3K79me3vsH3.YPD	75.83	75.56	75.69	**78.02**	**77.87**	**77.94**	75.50	75.37	75.44	77.81	77.73	77.77
	(76.24)	(76.02)	(76.13)	(78.95)	(78.31)	(78.63)	(76.44)	(76.21)	(76.32)	(78.03)	(77.38)	(77.70)

Note: 1.Pre., Rec., F1. are precision, recall and F1-measure, respectively.
2.The numbers in the brackets are prediction results by using SVM [15]

$$\begin{aligned} Precision_{positive} &= \frac{TP}{TP+FP}; Precision_{negative} = \frac{TN}{TN+FN} \\ Recall_{positive} &= \frac{TP}{TP+FN}; Recall_{negative} = \frac{TN}{TN+FP} \\ Precision &= \frac{Precision_{positive}+Precision_{negative}}{2} \\ Recall &= \frac{Recall_{positive}+Recall_{negative}}{2} \\ F1-measure &= \frac{2*(Precision*Recall)}{Precision+Recall} \end{aligned}$$

where TP, TN, FP, FN are the number of true positive, true negative, false positive and false negative examples, respectively.

Through various experiments we found that our method gave the best results when predicting nucleosome occupancy, acetylation, and methylation for DNA sequence areas of length $L = 500$ (data not shown). Due to the computational complexity, we have only tried with $k \leq 6$ and report here the results from sets of k-grams with k=5, k=6, k=4,5, and k=5,6.(Table 2).

The highest performance of our CRF method (at 18th L-BFGS iteration) for relative histone occupancy predictions (H3, H4, H3.H2O2), and acetylation predictions (H3K9acvsH3, H3K14acvsH3, H3K14acvsWCE, H3K14acvsH3.H2O2, H4acvsH3, H4acvsH3.H2O2), as well as methylation predictions (H3K4me1vsH3, H3K4me2vsH3, H3K4me3vsH3, H3K36me3vsH3, H3K79me3vsH3.YPD) achieved when we use features of both 5-grams and 6-grams (Table 2). The numbers in the brackets are the performance of the support vector machine (SVM)-based method (which was used in [15] to address the same problem) when using the same binary k-gram features. As it can be seen, CRF method is competitive with SVM-based method. In some cases, CRFs gave better performance, but in others performance was worse. SVM method can take into account the number of k-gram occurrences that represents DNA sequence better than binary k-gram features, hence SVM method can achive better performance [15]. However, CRFs have some advantages over SVMs such as they can easily incorporate knowledge into their prediction, and in the future we will take account annotated information concerning DNA sequence into our CRF method to improve the prediction results.

3.2 Genetic Area Preferences of Histone Occupancy, Acetylation, and Methylation

During the training CRFs model, we reported the weight of features (i.e. weight vector, see Section 2.2). In a CRF model, features with the larger weight would be more relevant than those with lower weight. We ranked the features based on their weight supporting for either positive or negative classes in CRF models, which were trained on 14 datasets. Table 3 and Table 4 show the most informative features from a set of 4-grams and 5-grams at 18th L-BFGS iteration (which did though give the best performance (Table 2), but to make later interpretation easily) for histone occupancy, acetylation, and methylation.

Informative features ranked by our CRF-based method agree with those from the previous SVM-based method [15]. They can be useful to analyze the genetic area preferences of histone occupancy, acetylation, and methylation. For example, CG (CpG) is a dinuceotide that appears very often in the most informative

Table 3. Most informative features selected from CRFs model for positive class with k-grams=4 and k-grams=5

Dataset	Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight
H3.YPD	CTTCA	0.16	CTTTA	0.15	TGCAG	0.14	ACAGC	0.14
	CGGC	0.13	TGAAG	0.13	GTTTG	0.13	GCGA	0.13
	GTGAT	0.13	TCATC	0.13	TGGC	0.13	CAGC	0.13
H4.YPD	TAAT	0.26	CTTCA	0.23	CAAAT	0.22	GCCAC	0.20
	GGATC	0.20	CTGGT	0.19	TTGGG	0.19	ATTTG	0.18
	ATCAG	0.18	GCAG	0.18	TATA	0.18	TTTA	0.17
H3.H2O2	CCGC	0.21	GCGC	0.21	CGGC	0.20	CGGG	0.19
	GCCG	0.18	CGCG	0.18	GGCC	0.18	CATGG	0.17
	CCGG	0.16	CCCT	0.15	CGCC	0.15	CCACC	0.15
H3K9acvsH3.YPD	CATGC	0.11	CAGGG	0.10	GTTCG	0.10	GCGAG	0.10
	CTTAG	0.09	TCTCG	0.09	TACC	0.09	GATAC	0.09
	CCCCG	0.09	AGGCG	0.09	GCCGG	0.09	CACCG	0.09
H3K14acvsH3.YPD	GCGTG	0.12	TTTTT	0.10	TAGTC	0.09	CTCGC	0.09
	CTCAT	0.09	CACC	0.08	TCTCT	0.08	ATATA	0.08
	CTTTT	0.08	AAAAA	0.08	AGCGG	0.08	TTTTC	0.08
H3K14acvsWCE.YPD	ACGGT	0.10	TCTCT	0.10	AGCCT	0.09	CTCAT	0.09
	CGGA	0.09	CGGC	0.09	CACC	0.09	TCCG	0.09
	AGTCG	0.08	TTGCT	0.08	ATGCG	0.08	GGAGT	0.08
H3K14acvsH3.H2O2	AGGGG	0.12	CCCCT	0.11	TAGTC	0.10	CACC	0.10
	CGAGG	0.09	CACAC	0.09	CGTAC	0.09	CCCGG	0.08
	ATGCG	0.08	TTAGT	0.08	TCTCT	0.08	CGTGC	0.08
H4acvsH3.YPD	CTCAT	0.12	AGCAA	0.10	CACAC	0.10	CACC	0.09
	GAAAA	0.09	GATAC	0.08	CATGC	0.08	TACCC	0.08
	TAGTC	0.08	TTAT	0.08	TCTCT	0.07	CAAGT	0.07
H4acvsH3.H2O2	AGGGG	0.18	GGGGG	0.14	AAAAG	0.13	CCCCT	0.12
	GTGGC	0.11	AAGGG	0.10	CTCCC	0.09	CTTGT	0.09
	ACACG	0.09	GATAC	0.09	GGGAG	0.09	CCTCG	0.08
H3K4me1vsH3.YPD	GGCA	0.08	TATC	0.08	CCAG	0.08	CTTGA	0.08
	TTAA	0.08	TGCGG	0.08	TGCAT	0.07	CCTCA	0.07
	TCCAA	0.07	AACCC	0.07	AGTT	0.07	GGTTG	0.07
H3K4me2vsH3.YPD	CTCAT	0.06	ATGAG	0.06	GGGAA	0.06	CTTGT	0.06
	AGACA	0.06	GATCT	0.05	CACTT	0.05	ACCAC	0.05
	AGTCC	0.05	GCTTA	0.05	AAAGA	0.05	GTCCA	0.05
H3K4me3vsH3.YPD	CACC	0.10	ACCCG	0.09	AGCCA	0.09	CAAGT	0.08
	GTCCA	0.08	GTCAA	0.08	TCTCT	0.08	GAAAA	0.07
	GCGTG	0.07	CTCAT	0.07	TAGTC	0.07	TCACT	0.07
H3K36me3vsH3.YPD	AAAA	0.14	TACT	0.12	ATAT	0.10	TTTT	0.10
	GTGA	0.10	CCTCC	0.09	TAAT	0.09	CGTCC	0.09
	CATCA	0.09	AGTT	0.09	AACA	0.09	GGACG	0.09
H3K79me3vsH3.YPD	TATA	0.22	TAAT	0.22	TAAA	0.16	ATAT	0.16
	TATT	0.14	ATTA	0.14	CATCA	0.14	TTAGA	0.14
	TGCA	0.13	TACT	0.13	TTTA	0.13	GATTT	0.11

negative features (Table 4). In other words, CG-rich DNA sequence areas are often free of histone occupancy, acetylation, or methylation. We all knew that CpG islands are usually near to gene starts. So we can infer from our results that promoter regions are often not occupied by nucleosomes. This is consistent with previous results by experimental approaches in vivo [16].

4 Conclusion

We have introduced a conditional model based method to predict qualitative histone occupancy, acetylation, and methylation areas in DNA sequences. We have selected a basic set of features based on DNA-sequence. Moreover, our model can evaluate the informative features to discriminate between DNA areas

Table 4. Most informative features selected from CRFs model for negative class with k-grams=4 and k-grams=5

Dataset	Feature	Weight	Feature	Weight	Feature	Weight	Feature	Weight
H3.YPD	CGCGC	0.15	TTTTT	0.13	AAAAA	0.12	GCGCG	0.11
	CGCGG	0.09	CCGCG	0.09	CGGGC	0.09	CGTGC	0.08
	GCGGG	0.08	TTATA	0.08	TTTTA	0.07	GGCCG	0.07
H4.YPD	AAAAA	0.38	TTTTT	0.29	AGAAA	0.27	GCGCG	0.27
	CGGAC	0.26	TTATA	0.26	TATAT	0.25	CGTGC	0.23
	CGCGC	0.23	CCCGG	0.22	GGCT	0.22	CGCGG	0.21
H3.H2O2	CGCGC	0.35	GCGCG	0.27	GCGGG	0.23	CGCGG	0.22
	CCGCG	0.22	TTTTT	0.20	AGGT	0.18	CTTC	0.16
	CCCCC	0.16	GGGCG	0.15	CCGGG	0.15	ACCA	0.14
H3K9acvsH3.YPD	GCCGC	0.13	GCAC	0.10	TCCAA	0.10	CCTCC	0.10
	ATTTG	0.09	AAAG	0.09	TTCTG	0.09	CAAAT	0.09
	TCTT	0.09	ATATT	0.09	GCAG	0.09	GCTG	0.09
H3K14acvsH3.YPD	GCCGC	0.11	CCAAT	0.09	TTATC	0.08	CTCGT	0.08
	ATTTG	0.08	ATTCA	0.08	TGATG	0.08	AAATT	0.08
	CCAAA	0.07	TCTAA	0.07	CATCA	0.07	TCAG	0.07
H3K14acvsWCE.YPD	CGCGG	0.14	GCCGC	0.12	AAGC	0.12	GCGGC	0.11
	CTTA	0.11	TCTT	0.10	CGCGC	0.10	TCAG	0.10
	AACA	0.10	CAAG	0.09	CTCT	0.09	GTCC	0.09
H3K14acvsH3.H2O2	AAATT	0.12	TAGT	0.11	TACG	0.08	GTGGG	0.08
	CTTA	0.08	TATTA	0.08	GCGTC	0.08	GTGA	0.08
	AAGC	0.08	CATA	0.08	TCCC	0.07	AATCA	0.07
H4acvsH3.YPD	GCCGC	0.15	TATTA	0.11	CTCT	0.11	GCGGC	0.10
	CAAG	0.09	TCGGA	0.09	TTATC	0.09	ATATT	0.08
	TTTGA	0.08	AACA	0.08	TTCTT	0.08	CCAAA	0.08
H4acvsH3.H2O2	TATTA	0.10	TCGT	0.09	TGGAT	0.09	ATATT	0.09
	TTTTG	0.09	TAATT	0.08	AATTT	0.08	ACAG	0.08
	AAGC	0.08	TACG	0.08	CCATA	0.08	TAAAA	0.08
H3K4me1vsH3.YPD	GAAG	0.10	TACAC	0.10	CCGAG	0.09	TATGT	0.08
	CAATT	0.08	ATAGT	0.08	CCGGC	0.08	CGAGG	0.08
	ACCCG	0.07	GCGTG	0.07	TGGG	0.07	TCCTA	0.07
H3K4me2vsH3.YPD	ATATT	0.09	TATTA	0.08	TGAAG	0.07	AATAT	0.06
	TAATA	0.06	TAATT	0.05	TTAAT	0.05	GTAAT	0.05
	CTAAA	0.05	GCCGC	0.05	AACAT	0.05	ATCAT	0.05
H3K4me3vsH3.YPD	GCCGC	0.14	CGCGG	0.09	CAAG	0.09	TCAG	0.09
	ACCCC	0.09	GCGGC	0.09	CTTA	0.09	AAGC	0.08
	GCGCG	0.08	CTCT	0.08	AACA	0.07	GCGTC	0.07
H3K36me3vsH3.YPD	GAAG	0.18	ATAGT	0.12	TATAT	0.12	AAAAG	0.11
	TAGGA	0.10	CTTAA	0.10	CTCGA	0.09	CACC	0.09
	CTCAT	0.09	GTACT	0.09	ACCCG	0.09	ACATA	0.09
H3K79me3vsH3.YPD	TATAT	0.22	ATATA	0.20	ACATA	0.19	AAAAA	0.17
	TTGT	0.16	TTATA	0.16	TATAA	0.16	GCCGC	0.15
	ACGTA	0.13	GAAG	0.13	GCCCG	0.13	CTTC	0.12

with high and low occupancy, acetylation, or methylation. In the near future, we plan to incorporate features related to sequence motifs into our method in order to capture more faithfully the constrains on the model.

Acknowledgements

The research described in this paper was partially supported by the Institute for Bioinformatics Research and Development of the Japan Science and Technology Agency, and by COE project JCP KS1 of the Japan Advanced Institute of Science and Technology. We also would like to thank Hieu P.X. for providing FlexCRFs package and sharing with us his experience in machine learning area.

References

1. B. E. Bernstein, E. L. Humphrey, R. L. Erlich, R. Schneider, P. Bouman, J. S. Liu, T. Kouzarides, and S. L. Schreiber. Methylation of histone H3 Lys 4 in coding regions of active genes. *Proc. Natl. Acad. Sci. USA*, 99(13):8695–8700, 2002.
2. B. E. Bernstein, C. L. Liu, E. L. Humphrey, E. O. Perlstein, and S. L. Schreiber. Global nucleosome occupancy in yeast. *Genome Biol.*, 5(9):R62, 2004.
3. S. F. Chen and R. Rosenfeld. *A gaussian prior for smoothing maximum entropy models.* Technical report CMU-CS-99-108, 1999.
4. T. Kouzarides. Histone methylation in transcriptional control. *Curr. Opin. Genet. Dev.*, 12(2):198–209, 2002.
5. S. K. Kurdistani, S. Tavazoie, and M. Grunstein. Mapping global histone acetylation patterns to gene expression. *Cell*, 117(6):721–733, 2004.
6. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conference on Machine Learing*, 2001.
7. C. K. Lee, Y. Shibata, B. Rao, B. D. Strahl, and J. D. Lieb. Evidence for nucleosome depletion at active regulatory regions genome-wide. *Nat. Genet.*, 36(8):900–905, 2004.
8. D. Liu and J.Nocedal. On the limited memory bfgs method for large-scale optimization. *Mathematical Programming*, 45:503–528, 1989.
9. K. Luger, A. W. Mader, R. K. Richmond, D. F. Sargent, and T. J. Richmond. Crystal structure of the nucleosome core particle at 2.8 A resolution. *Nature*, 389(6648):251–260, 1997.
10. R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. Proceeding CoNLL*, 2002.
11. A. McCallum. Maximum entropy markov models for information extraction and segmentation. In *Proc. 15th International Conference on Machine Learing*, 2000.
12. A. McCallum. Efficiently inducing features of conditional random fields. In *Proc. 19th Conference on Uncertainy in Artificial Intelligence*, 2003.
13. G. J. Narlikar, H. Y. Fan, and R. E. Kingston. Cooperation between complexes that regulate chromatin structure and transcription. *Cell*, 108(4):475–487, 2002.
14. C. L. Peterson and M. A. Laniel. Histones and histone modifications. *Curr. Biol.*, 14(14):R546–R551, 2004.
15. T.H. Pham, D.H. Tran, T.B. Ho, K. Satou, and G. Valiente. Qualitatively predicting acetylation and methylation areas in dna sequences. In *Proc. 16th International Conference on Genome Informatics*, 2005.
16. D. K. Pokholok, C. T. Harbison, S. Levine, M. Cole, N. M. Hannett, T. I. Lee, G. W. Bell, K. Walker, P. A. Rolfe, E. Herbolsheimer, J. Zeitlinger, F. Lewitter, D. K. Gifford, and R. A. Young. Genome-wide map of nucleosome acetylation and methylation in yeast. *Cell*, 122(4):517–527, 2005.
17. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. Proceeding of IEEE*, pages 257–286, 1989.
18. B. Ren, F. Robert, and et al. Genome-wide location and function of DNA binding proteins. *Science*, 290(5500):2306–2309, 2000.
19. D. Robyr, Y. Suka, I. Xenarios, S. K. Kurdistani, A. Wang, N. Suka, and M. Grunstein. Microarray deacetylation maps determine genome-wide functions for yeast histone deacetylases. *Cell*, 109(4):437–446, 2002.
20. F. Sha and F. Pereira. Shalow parsing with conditional random fields. In *Proc. 15th Proceeding of Human Language Technology*, 2003.
21. H. Wallach. *Efficient Training of Conditional Random Fields.* Master thesis, 2002.

DNA Fragment Assembly: An Ant Colony System Approach

Wannasak Wetcharaporn[1], Nachol Chaiyaratana[1,2], and Sissades Tongsima[3]

[1] Research and Development Center for Intelligent Systems,
King Mongkut's Institute of Technology North Bangkok,
1518 Piboolsongkram Road, Bangsue, Bangkok 10800, Thailand
w_wannasak@hotmail.com, nchl@kmitnb.ac.th
[2] Institute of Field Robotics,
King Mongkut's University of Technology Thonburi,
91 Pracha u-tid Road, Bangmod, Thungkru, Bangkok 10140, Thailand
[3] National Center for Genetic Engineering and Biotechnology,
National Science and Technology Development Agency,
113 Thailand Science Park, Phahonyothin Road, Pathumthani 12120, Thailand
sissades@biotec.or.th

Abstract. This paper presents the use of an ant colony system (ACS) algorithm in DNA fragment assembly. The assembly problem generally arises during the sequencing of large strands of DNA where the strands are needed to be shotgun-replicated and broken into fragments that are small enough for sequencing. The assembly problem can thus be classified as a combinatorial optimisation problem where the aim is to find the right order of each fragment in the ordering sequence that leads to the formation of a consensus sequence that truly reflects the original DNA strands. The assembly procedure proposed is composed of two stages: fragment assembly and contiguous sequence (contig) assembly. In the fragment assembly stage, a possible alignment between fragments is determined with the use of a Smith-Waterman algorithm where the fragment ordering sequence is created using the ACS algorithm. The resulting contigs are then assembled together using a nearest neighbour heuristic (NNH) rule. The results indicate that in overall the performance of the combined ACS/NNH technique is superior to that of the NNH search and a CAP3 program. The results also reveal that the solutions produced by the CAP3 program contain a higher number of contigs than the solutions produced by the proposed technique. In addition, the quality of the combined ACS/NNH solutions is higher than that of the CAP3 solutions when the problem size is large.

1 Introduction

In order to understand the whole genetic makeup of an organism, the information regarding the entire DNA (deoxyribonucleic acid) sequence is required. The most famous research project that attempts to attain such information is the Human Genome Project [1] where the entire DNA sequence of a human genome, covering over three billion genetic codes, is investigated. To achieve the goal, the project has to be divided into many components. One of major components is DNA fragment assembly. DNA is a double helix comprised of two complementary strands of polynucleotides. Each strand of DNA can be viewed as a character string over an alphabet of four letters: A, G, C and T. The four letters represent four bases, which are adenine (A), guanine (G), cytosine (C) and thymine (T). The two strands are complementary in the sense that at corresponding positions A's are always paired with T's and C's with G's. These pairs of complementary bases are referred to as "base pairs". At present, strands of DNA that are longer than 600 base pairs cannot routinely be sequenced accurately [2]. The sequencing technique that involves fragmentation of a DNA strand is called a shotgun sequencing technique. Basically, DNA is first replicated many times and then individual strands of the double helix are broken randomly into smaller fragments. This produces a set of *out of order* fragments short enough for sequencing.

A DNA fragment assembly problem involves finding the right order of each fragment in the fragment ordering sequence, which leads to the formation of a consensus sequence that truly reflects the parent DNA strands. In other words, the DNA fragment assembly problem can be treated as a combinatorial optimisation problem. A number of deterministic and stochastic search techniques have been used to solve DNA fragment assembly problems [3]. For instance, Huang and Madan [4] and Green [5] have used a greedy search algorithm to solve the problem. However, a manual manipulation on the computer-generated result is required to obtain a biologically plausible final result. Other deterministic search algorithms that have been investigated include a branch-and-cut algorithm [6] and a graph-theoretic algorithm where DNA fragments are either represented by graph nodes [7, 8] or graph edges [9]. The capability of stochastic search algorithms such as a simulated annealing algorithm [10], a genetic algorithm [11, 12, 13] and a neural network based prediction technique [14] has also been investigated. The best DNA fragment assembly results obtained from stochastic searches have been reported in Parsons and Johnson [12], and Kim and Mohan [13] where genetic algorithms have proven to outperform greedy search techniques in relatively small-sized problems. In addition, the need for manual intervention is also eliminated in this case. Although a significant improvement over the greedy search result has been achieved, the search efficiency could be further improved if the redundancy in the solution representation could be eliminated from the search algorithms [11]. Similar to a number of combinatorial optimisation techniques, the use of a permutation representation is required to represent a DNA fragment ordering solution in a genetic algorithm search. With such representation, different ordering solutions can produce the same DNA consensus sequence. Due to the nature of a genetic algorithm as a parallel search technique, the representation redundancy mentioned would inevitably reduce the algorithm efficiency. A stochastic search algorithm that does not suffer from such effect is an ant colony system (ACS) algorithm [15].

The natural metaphor on which ant algorithms are based is that of ant colonies. Real ants are capable of finding the shortest path between a food source and their nest without using visual clues by exploiting pheromone information. While walking, ants deposit pheromone on the ground, and probabilistically follow pheromone previously deposited by other ants. The way ants exploit pheromone to find the shortest path between two points can be described as follows. Consider a situation where ants arrive at a decision point in which they have to decide between two possible paths for both getting to and returning from their destination. Since they have no clue about which is the best choice, they have to pick the path randomly. It can be expected that on average half of the ants will decide to go on one path and the rest choose to travel on the other path. Suppose that all ants walk at approximately the same speed, the pheromone deposited will accumulate faster on the shorter path. After a short transitory period the difference between the amounts of pheromone on the two paths is sufficiently large so as to influence the decision of other ants arriving at the decision point. New ants will thus prefer to choose the shorter path since at the decision point they perceive more pheromone. At the end, all ants will use the shorter path. If the ants have to complete a circular tour covering n different destinations without visiting order preference, the emerged shortest path will be a solution to the n-city travelling salesman problem (TSP). Although the ACS algorithm also exploits stochastic parallel search mechanisms, the algorithm performance does not depend upon the solution representation. This is because the optimal solution found by the ACS algorithm will emerge as a single "shortest path". In other words, the problem regarding the redundancy in the solution representation mentioned early would be completely irrelevant to the context of the ACS algorithm search.

The organisation of this paper is as follows. In section 2, the overview of a DNA fragment assembly problem will be given. In section 3, the background on the ACS algorithm will be discussed. The application of the ACS algorithm on the DNA fragment assembly problem will be explained in section 4. Next, the case studies are explained in section 5. The results obtained after applying the ACS algorithm to the problem and the result discussions are given in section 6. Finally, the conclusions are drawn in section 7.

2 DNA Fragment Assembly Problem

In order to create a DNA map, the original complementary strands of DNA are first replicated many times. Then individual strands of the double helix are broken randomly into small fragments. Base ordering in each fragment can subsequently be identified by applying a base calling procedure on the fluorescent trace-data [16]. After the bases on each fragment have been sequenced, the fragments are then aligned in order to create a consensus sequence that represents the original or parent DNA strands. An alignment between two fragments can be created if there is a portion from each fragment that together can produce a match between either the same-base ordering sequences or the complementary-base ordering sequences. This is because the fragments can come from the same strand or different complementary strands. The alignments of fragments are

```
                                    Parent Strands
                                  TTAGCACAGGAACTCTA
                                  |||||||||||||||||
                                  AATCGTGTCCTTGAGAT

Original Fragment Set      DNA Fragment Assembly      Consensus Sequence
AGCAC                      TTTGC-C                    TTAGCACAGGAACTCTA
ATCAAGGAAC                   AGCAC
GACTC                        ATCA-AGGAAC
TTCTA                               GA-CTC
TTTGCC                                 TTCTA
```

Fig. 1. Schematic diagram of the fragment assembly process

schematically displayed in Fig. 1. From Fig. 1, the consensus sequence truly represents a strand from the parent DNA strands because there is an overlap portion between each pair of aligned fragments. The number of matching bases between two aligned fragments together with penalties from mismatches and gaps are generally referred to as the overlap score. In order to obtain an overlap score, the alignment are chosen to maximise the number of matching bases between the two fragments and minimise mismatches and gaps. If the search for a possible alignment between a given fragment and other fragments returns either a relatively low or a zero overlap score, there will be gaps in the consensus sequence. In such a case, the consensus sequence will contain multiple disjoint sequences called contiguous sequences or contigs. In other words, it is desirable to have only one contig in the consensus sequence. Notice that each base on the consensus sequence is determined by applying a majority vote rule to each column of bases in the aligned fragments.

3 Ant Colony System Algorithm

The ant colony system (ACS) algorithm is a search algorithm, which has its root from the study of insect collective behaviour [15]. The search algorithm is suitable to a combinatorial optimisation problem, which has a characteristic similar to that of a TSP. In the context of a TSP, an ant will choose to make a transition from one city to the next city using the information regarding the distance and the pheromone deposited between the cities. Pheromone is a substance left on the path by other ants that have previously made a transition between the two cities of interest. The level of pheromone is directly correlated to the number of ants that have travelled between the cities and hence inversely proportional to the distance between the cities. The more pheromone being deposited on the path, the higher the number of ants that will choose to make the transition through that path. After all ants have completed the circular tour that covers all cities, the paths on the shortest global tour will receive additional pheromone deposition. This will encourage more ants to make a tour that utilises paths, which make up the shortest global tour in the future. By reinforcing the decision that leads to the construction of the shortest global tour using pheromone deposition, an optimal solution to the TSP would be co-operatively created by all ants in the colony.

Based on the overview of the algorithm, three main components that make up the algorithm are a state transition rule, a local pheromone-updating rule and a global pheromone-updating rule. In addition, Dorigo and Gambardella [15] have also introduced the use of a data set called a candidate list that creates a limitation on the city chosen by an ant for a state transition. The explanation on these three rules and the candidate list can be found in Dorigo and Gambardella [15]. Although the explanation of the ACS algorithm is given in the context of a TSP, the ACS algorithm can be easily applied to a DNA fragment assembly problem. This is because the overlap score, which provides information regarding how well two fragments can fit together, can be directly viewed as the inverse of the distance between two cities.

4 Application of the ACS Algorithm to the DNA Fragment Assembly Problem

A DNA fragment assembly problem can generally be viewed as a combinatorial optimisation problem that is closely related to a TSP [11]. Detailed comparison between the two problems and how the ACS algorithm can be applied to the DNA fragment assembly problem are discussed as follows.

4.1 Comparison Between a DNA Fragment Assembly Problem and a TSP

By making an analogy between cities in a TSP and fragments in an assembly problem, it can be easily seen that the overlap score can be viewed as the inverse of the distance between cities. However, a DNA fragment assembly problem is a special kind of symmetric TSPs. In brief, distances between cities r and s in the forward and backward journeys are equal in a symmetric TSP. A factor that makes a DNA fragment assembly problem a special form of symmetric TSPs is the consideration on the original parent DNA strand at which the fragment came. Each fragment used in the assembly process has an equal probability of coming from one of the two parent DNA strands. With different assumptions on the origin of the fragment, the resulting overlap score would also be different. With this factor, there are four possible configurations for obtaining the overlap score between two fragments. The summary of four alignment configurations is given in Table 1. From Table 1, during the use of the ACS algorithm if an ant is at fragment r where the fragment is assumed to come from the forward DNA strand, the only possible configurations for an alignment with fragment s are configurations 1 and 2. On the other hand, if the fragment r comes from the complementary strand where the order of base reads must always be in reverse, the feasible configurations for an alignment with fragment s are configurations 3 and 4.

Table 1. Four alignment configurations between two fragments

Configuration	Assumption about the Strand of Origin	
	Fragment r	Fragment s
1	Forward	Forward
2	Forward	Reverse complement
3	Reverse complement	Forward
4	Reverse complement	Reverse complement

4.2 Evaluation of an Overlap Score

From the previous sub-section, there are four possible configurations at which two fragments can be aligned. An overlap score for an alignment between two fragments can be calculated using a Smith-Waterman algorithm [17]. In brief, the algorithm computes the locally best alignment between two fragments using a dynamic programming approach. The only parameter settings required for the Smith-Waterman algorithm are the similarity score matrix and the affine gap penalty. In this paper, the similarity score matrix and the affine gap penalty settings are based on the default settings of a PHRAP assembly program [5]. Basically, the scores $+1$ and -9 are used for a match and a mismatch involving A, C, G or T, respectively while the score 0 is used for a match or a mismatch involving N—no base calling character. In addition, the score -11 is used for the gap opening penalty (the first residue in a gap) and the score -10 is used for the gap extension penalty (each subsequent residue). In this paper the SIM program [18], which is an implementation of the Smith-Waterman algorithm in a C programming language, is used to obtain the overlap score between two fragments. With the use of the Smith-Waterman algorithm, the configurations 1 and 4 in Table 1 will have the same overlap score since both fragments are assumed to come from the same parent strand. Similarly, the configurations 2 and 3 in Table 1 will have the same overlap score since the fragments are assumed to come from different strands. After the overlap score has been calculated in this manner, the fragment ordering sequence that leads to the formation of a consensus sequence can be constructed by adding fragments to the existing ordering sequence by one fragment at a time. In a schematic display, the fragment ordering sequence would look similar to a ladder. This results from the way the relative position of the fragment in the alignment is dictated by the overlap score calculation procedure.

4.3 ACS Search Objective

The objective function investigated is a minimisation function, which is a combination between the number of contigs and the difference in length between the longest and the shortest contigs. With the use of this objective function, the solution that has the lesser number of contigs will be regarded as the better solution. The locations of the beginning and the end of each contig in the fragment ordering sequence are the locations where the overlap score between two consecutive fragments is lower than a threshold value, which in this investigation is set to 95. However, more than one solution generated may have the same number of contigs. If this is the case, the solution that is the better solution is the one where the difference between the length of the longest and the shortest contigs is minimal. This part of the objective function is derived from the desire that the ultimate goal solution is the one with either only one contig or the fewest possible number of contigs where each contig is reasonably long.

4.4 ACS State Transition and Pheromone-Updating Rules

In order to use an ACS algorithm in a DNA fragment assembly process, the terms in the ACS rules that are required to change are the inverse of distance term and the pheromone-updating factors. In the ACS state transition rule, a decision is made using guidance from a pheromone-closeness product. The inverse of distance terms can thus change to the overlap score between two fragments. Similarly, in the ACS pheromone-updating rules, the tour length used in the pheromone-updating factor ($\Delta\tau$) will be substituted by the sum of inverse of overlap scores obtained for the fragment ordering sequence generated. In the ACS local pheromone updating rule, the pheromone updating factor $\Delta\tau(r, s)$ will be set to $\Delta\tau(r, s) = \tau_0 = (n\, IS_{nn})^{-1}$ where IS_{nn} is the sum of inverse of overlap scores obtained using a nearest neighbour heuristic (NNH) rule. On the other hand, the pheromone-updating factor $\Delta\tau(r, s)$ in the ACS global pheromone updating rule will be set to $\Delta\tau(r, s) = (IS_{gb})^{-1}$ for $(r, s) \in$ global-best-solution where IS_{gb} is the sum of inverse of overlap scores for the globally best solution. The final ACS consensus sequence will be obtained by splitting the circular fragment ordering sequence at the location where the overlap score between two fragments is minimal.

5 Case Studies

The capability of the ACS algorithm in DNA fragment assembly will be tested using data sets obtained from a GenBank database at the National Center for Biotechnology Information or NCBI (`http://www.ncbi.nlm.nih.gov`). The parent DNA strands in this case are extracted from the human chromosome 3 where the strands with the sequence length ranging from 21K to 83K base pairs are utilised. It is noted that each complementary pair of parent DNA strands can be referred from the database using its accession number. The fragments used to construct the consensus sequence are also obtained from the same database where each fragment is unclipped (low quality base reads are retained) and has the total number of bases between 700 and 900. This means that the fragments used in the case studies contain sequencing errors generally found in any experiments. In addition, no base quality information is used during the assembly investigation. In order for a consensus sequence to accurately represent the parent DNA strands, there must be more than one fragment covering any base pairs

Table 2. Information on the data set

Accession Number	AC023501		AC023159		AC005903		AC026318	
Base Pair	20,824		34,680		63,949		83,181	
Case Study	1	2	3	4	5	6	7	8
Coverage	10	10	5	5	7	6	7	7
Number of Fragments	368	367	279	269	611	591	709	708
Gaps	0	1	0	6	0	1	0	1

of the parent strands. The average number of fragments covering each base pair on the parent strands is generally referred to as coverage. In addition, for a consensus sequence to be made up from one contig, there must be no gaps in the fragment ordering sequence. In this paper, the data set is prepared such that the consensus sequence contains either one contig or multiple contigs. The summary of the data set descriptions is given in Table 2.

In order to benchmark the performance of the ACS algorithm, its search performance will be compared with that of the NNH rule and a CAP3 program [4], which is one of the most widely used programs in bioinformatics research community. Since the base quality information is not used during the assembly, the solutions produced by the CAP3 program would be similar to the results from a PHRAP program [5], which is also a standard program [13]. The parameter setting for the ACS algorithm is the recommended setting for solving symmetric travelling salesman problems given in Dorigo and Gambardella [15].

6 Results and Discussions

The ACS algorithm, the NNH rule and the CAP3 program have been applied to all eight case studies. In the case of the NNH search, all possible n solutions with different starting fragments are generated where n is the number of fragments. The solutions are obtained using the sum of overlap scores as the maximisation objective. The best solution is then picked where contigs are produced by assembling aligned fragments together and applying a majority-vote rule, as illustrated in Fig. 1, for the base calling purpose. Next, an attempt on DNA contig assembly is made where the overlap score between each contig is obtained using a Smith-Waterman algorithm and the NNH rule is subsequently applied. Similar to the early assembly procedure, all possible l solutions are generated this time where l is the number of contigs from the primary assembly stage and the best solution among l solutions are chosen as the final solution. In contrast, the ACS algorithm runs are repeated ten times in each case study using the optimisation objective explained in sub-section 4.3. During each run, the initial solution used is randomly chosen from all n solutions produced by the NNH rule. Similar to the case of the NNH rule, after all ACS runs are finished, the best solution in terms of the search objective employed is picked and contigs are obtained by assembling fragments together. The contig assembly is then commenced where the NNH rule is applied using the sum of overlap scores as the maximisation objective. It is noted that since the CAP3 program is deterministic in nature, the program is executed only one time for each case study. From the assembly results obtained, two discussion topics can be given: the number of contigs in the final assembly solutions and the quality of the final solutions. These issues are discussed as follows.

6.1 Number of Contigs in the Final Assembly Solutions

As mentioned earlier, after the primary DNA fragment assembly stage where either the NNH rule or the ACS algorithm is applied to the problem, the resulting DNA contigs are subsequently assembled together using the NNH rule. The final numbers of contigs obtained from both approaches—the NNH+NNH approach and the ACS+NNH approach—together with the number of contigs from the solution generated by the CAP3 program, are reported in Table 3. From Table 3, the CAP3 program outperforms the ACS+NNH approach in cases 4 and 5 while the ACS+NNH approach is the best technique in the remaining cases. The results also indicate that as the problem size increases, the number of contigs produced by the ACS+NNH approach also increases. On the other hand, it appears that there is no correlation between the problem size and the number of contigs in the case of the CAP3 program. Based upon the above observation, it is sufficed to say that the overall performance of the ACS+NNH approach is higher than that of the CAP3 program. The comparison between the performances of NNH+NNH and ACS+NNH approaches is now considered. Both techniques have the same performance in cases 1, 2, 7 and 8 while the ACS+NNH approach has a higher performance in cases 3, 4, 5 and 6. In overall, it can be concluded that there is a range on the problem size at which the ACS+NNH approach is capable of producing a better result than the NNH+NNH approach.

Table 3. Number of contigs from the solutions produced by the NNH+NNH approach, the ACS+NNH approach and the CAP3 program

Problem	Number of Contigs			
	Parent Strand	NNH+NNH	ACS+NNH	CAP3
AC023501 (21K bases)				
No gaps	1	1	1	3
With gaps	2	2	2	4
AC023159 (35K bases)				
No gaps	1	11	5	10
With gaps	7	18	11	9
AC005903 (64K bases)				
No gaps	1	14	11	3
With gaps	2	14	2	3
AC026318 (83K bases)				
No gaps	1	15	15	25
With gaps	2	15	15	25

6.2 Quality of the Assembly Solutions

In the previous sub-section, the numbers of contigs in the final assembly solutions from all three techniques are compared. In this part of the discussion, the quality of these contigs is examined. The quality of a contig is measured in terms of the base difference between the parent DNA sequence and the contig of interest using a Smith-Waterman algorithm with the setting described in sub-section 4.2. This difference can be expressed in terms of an assembly error, which can be further divided into three components: a substitution error, an insertion/deletion (indel) error and a coverage error. A substitution error appears when a base in one of two aligned sequences—a parent DNA sequence and a contig in this case—does not match its counterpart in the other sequence. When a base in one aligned sequence seems to have been deleted as the result of a divergence of the sequence from its counterpart, such absence is labelled as a deletion error in the derived sequence. On the other hand, when a base appears to have been inserted to produce a longer sequence, an insertion error is labelled in the augmented sequence. A deletion in one sequence can thus be viewed as an insertion in the other sequence. Hence, these two types of error are generally referred to together as an indel error. In contrast to substitution and indel errors, a coverage error is detected when there are bases in the parent DNA sequence, which are located outside the part of contig that best matches the parent sequence and thus not covered by any contigs. The assembly errors in the contigs produced by all three techniques, expressed in terms of the percentage of errors out of the total number of bases in the parent sequence, are compared in Table 4.

Basically, the sum of substitution and insertion/deletion errors from the ACS+NNH approach is higher than that of the CAP3 program in all case studies. However, in the first four case studies the coverage errors from the CAP3 program are either lower than or equal to that from the ACS+NNH approach while the solutions that have lower coverage errors in the last four case studies are produced by the ACS+NNH approach. It is also noticeable that the errors from the NNH+NNH and ACS+NNH approaches are very similar in all case studies except for the last two cases where the coverage errors of the solutions from the ACS+NNH approach are lower. Based upon the results in terms of solution quality, the right combination between the ACS algorithm and the CAP3 program may yield contigs that have even lower assembly errors. Since the core search algorithm of the CAP3 program is a greedy search algorithm [4], the replacement of the CAP3 core algorithm with the ACS algorithm would be one possible step towards the goal.

7 Conclusions

In this paper, a DNA fragment assembly problem, which is a complex combinatorial optimisation problem that can be treated as a travelling salesman problem (TSP), has been discussed. In the context of a TSP, a fragment ordering sequence would represent a tour that covers all cities while the overlap score between two aligned fragments in the ordering sequence can be viewed as the inverse of the

Table 4. Assembly errors expressed in terms of the sum of substitution and insertion/deletion errors, and the coverage error

Problem	Substitution & Indel Errors (%)			Coverage Error (%)		
	NNH+NNH	ACS+NNH	CAP3	NNH+NNH	ACS+NNH	CAP3
AC023501 (21K bases)						
No gaps	1.63	1.89	0.13	0.00	0.00	0.00
With gaps	1.62	1.43	0.10	0.00	0.19	0.00
AC023159 (35K bases)						
No gaps	1.17	1.57	0.21	8.48	8.22	6.98
With gaps	N/A	N/A	0.22	N/A	N/A	1.10
AC005903 (64K bases)						
No gaps	1.08	1.01	0.10	0.45	0.45	2.38
With gaps	1.02	0.97	0.11	0.47	1.22	2.06
AC026318 (83K bases)						
No gaps	1.19	1.11	0.39	12.42	7.44	11.08
With gaps	1.19	1.09	0.40	12.23	7.49	10.87

distance between two cities. The assembly procedure proposed consists of two stages: fragment assembly and contig assembly stages. In the fragment assembly stage, a search for the best alignment between fragments is carried out using an ant colony system (ACS) algorithm. The resulting contigs are then assembled together using a nearest neighbour heuristic (NNH) rule in the contig assembly stage. The assembly procedure proposed has been benchmarked against a CAP3 program [4]. The results suggest that the solutions produced by the CAP3 program contain a higher number of contigs than the solutions generated by the proposed technique. In addition, the quality of the combined ACS/NNH solutions is higher than that of the CAP3 solutions when the problem size is large. Since the core algorithm of the CAP3 program is a greedy search algorithm, a replacement of the core algorithm with the ACS algorithm may yield an improvement on the final assembly solution.

Acknowledgements

This work was supported by the Thailand Toray Science Foundation (TTSF) and National Science and Technology Development Agency (NSTDA) under the Thailand Graduate Institute of Science and Technology (TGIST) programme. The authors acknowledge Prof. Xiaogiu Huang at the Iowa State University for providing an access to the SIM and CAP3 programs.

References

1. International Human Genome Sequencing Consortium: Initial sequencing and analysis of the human genome. Nature **409**(6822) (2001) 860–921
2. Applewhite, A.: Mining the genome. IEEE Spectrum **39**(4) (2002) 69–71

3. Pop, M., Salzberg, S.L., Shumway, M.: Genome sequence assembly: Algorithms and issues. Computer **35**(7) (2002) 47–54
4. Huang, X., Madan, A.: CAP3: A DNA sequence assembly program. Genome Research **9**(9) (1999) 868–877
5. Green, P.: Phrap documentation. Phred, Phrap, and Consed **www.phrap.org** (2004)
6. Ferreira, C.E., de Souza, C.C., Wakabayashi, Y.: Rearrangement of DNA fragments: A branch-and-cut algorithm. Discrete Applied Mathematics **116**(1-2) (2002) 161–177
7. Batzoglou, S., Jaffe, D., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J.P., Lander, E.S.: ARACHNE: A whole-genome shotgun assembler. Genome Research **12**(1) (2002) 177–189
8. Kececioglu, J.D., Myers, E.W.: Combinatorial algorithms for DNA sequence assembly. Algorithmica **13**(1-2) (1995) 7–51
9. Pevzner, P.A., Tang, H., Waterman, M.S.: An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences of the United States of America **98**(17) (2001) 9748–9753
10. Burks, C., Engle, M., Forrest, S., Parsons, R., Soderlund, C., Stolorz, P.: Stochastic optimization tools for genomic sequence assembly. In: Adams, M.D., Fields, C., Venter, J.C. (eds.): Automated DNA Sequencing and Analysis. Academic Press, London, UK (1994) 249–259
11. Parsons, R.J., Forrest, S., Burks, C.: Genetic algorithms, operators, and DNA fragment assembly. Machine Learning **21**(1-2) (1995) 11–33
12. Parsons, R.J., Johnson, M.E.: A case study in experimental design applied to genetic algorithms with applications to DNA sequence assembly. American Journal of Mathematical and Management Sciences **17**(3-4) (1997) 369–396
13. Kim, K., Mohan, C.K.: Parallel hierarchical adaptive genetic algorithm for fragment assembly. In: Proceedings of the 2003 Congress on Evolutionary Computation, Canberra, Australia (2003) 600–607
14. Angeleri, E., Apolloni, B., de Falco, D., Grandi, L.: DNA fragment assembly using neural prediction techniques. International Journal of Neural Systems **9**(6) (1999) 523–544
15. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 53–66
16. Allex, C.F., Baldwin, S.F., Shavlik, J.W., Blattner, F.R.: Improving the quality of automatic DNA sequence assembly using fluorescent trace-data classifications. In: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, St. Louis, MO (1996) 3–14
17. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology **147**(1) (1981) 195–197
18. Huang, X., Miller, W.: A time-efficient, linear-space local similarity algorithm. Advances in Applied Mathematics **12**(3) (1991) 337–357

BeeHiveGuard: A Step Towards Secure Nature Inspired Routing Algorithms

Horst F. Wedde, Constantin Timm, and Muddassar Farooq

Informatik III, University of Dortmund,
44221 Dortmund, Germany
{wedde, timm, farooq}@ls3.cs.uni-dortmund.de

Abstract. Nature inspired routing protocols for fixed and mobile networks are becoming an active area of research. However, analyzing their security threats and countering them have received little attention. In this paper we discuss the security threats of a state-of-the-art routing protocol, *BeeHive*, and then extend the algorithm with our security model to counter them. We further conclude from our extensive experiments that standard cryptography techniques can not be utilized due to their large processing and communication costs, if Nature inspired routing protocols are to be deployed in real world networks.

1 Introduction

Nature inspired routing protocols are becoming an active area of research because they do not require an a priori global system model of the network rather they utilize a local system model as observed by the agents. The agents gather the network state in a decentralized fashion and leave the corresponding information on visited nodes. This information enables them to make routing decisions in a decentralized fashion without the need of having access to complete network topology. The algorithms can adapt autonomously to changes in the network, or in traffic patterns. *AntNet* [1], *BeeHive* [15] and *Distributed Genetic Algorithm (DGA)* [7] are state-of-the-art Nature inspired routing algorithms.

In all of the above-mentioned algorithms, the authors always implicitly trusted the identity of the agents and their routing information. However, this assumption is not valid for real world networks, where malicious intruders or compromised nodes can wreak havoc. To our knowledge, little attention has been paid to analyzing the security threats of Nature inspired routing protocols and efficiently countering them. The router vendors are not willing to deploy Nature inspired routing protocols in real networks because their security threats are not properly investigated. We believe that a scalable security framework, which has acceptable processing and communication costs, is an important step toward deployment of such protocols in real world routers. This observation provided the motivation for our current work in which we try to take the first step in this direction by doing a comprehensive analysis of the security threats of the *BeeHive* algorithm. We provide the important features of our security framework, *BeeHiveGuard*, and measure its processing and communication costs.

According to Perlman [9], a routing protocol can have anomalous behavior because of two types of failures: simple and Byzantine. Simple failures occur once a node crashes or a link goes down while Byzantine failures happen due to the malicious nodes that launch agents into the networks. Such agents can significantly alter the routing behavior of a routing protocol. In this work, we focus on Byzantine failures because *BeeHive* is resilient to simple failures [15].

The rest of the paper is organized as follows. In Section 2, we provide a brief overview of Byzantine failures/attacks that are relevant to Nature inspired routing protocols. We then outline important features of considered *BeeHive* algorithm in Section 3 and our security enhancements to it are discussed in Section 4. In Section 5, we discuss the simulated threat scenarios and then provide results from the extensive simulations in Section 6. Finally we conclude the paper with an outlook to our future research.

2 Security Challenges in Nature Inspired Routing Algorithms

In [6], the authors listed a number of attacks that the malicious nodes can launch in Mobile Ad Hoc Networks (MANETs). We briefly describe only those attacks that are relevant to Nature inspired routing protocols within the context of fixed networks.

- *Fabrication attacks* are launched by a router to change the normal route of a data packet. This is accomplished by retransmitting old agents or by modifying the information of agents or by launching bogus agents. A *fabrication attack* can be further classified as:
  - An *update storm or malicious flooding.* In this attack a malicious router injects a large number of agents in a short interval of time into the network. As a result, the information (mostly bogus) carried by its agents spreads faster in the network than the true information of other routers. Consequently, the malicious router can divert data packets towards itself.
  - A *replay attack.* In this attack a router retransmits old agents that carry outdated information in the network.
  - A *rushing attack.* This attack in only possible in those routing protocols in which the agents are identified with a unique sequence number. An attacker launches the agents whose source address is of some other node. Moreover, it assigns them a significantly high sequence number. In this way it forces other routers to accept its bogus agents and drop the real ones.
- *Dropping attacks* are powerful because they can divide a network into several partitions. They are of two types: blackhole attack and network partition.
  - *Blackhole attack.* In this attack, an attacker diverts data packets towards itself and simply drops them.
  - *Network partition.* An attacker tries to separate a network into k ($k \geq 2$) partitions. As a result, the nodes in one partition can not communicate with the nodes in the other partitions.

– *Tampering attack.* In this attack, a malicious node simply modifies the routing information carried by an agent to its own benefits.
  - *Identity impersonating or spoofing.* In this attack, a router impersonates another router by launching bogus agents. As a result, the malicious router can force data packets not to follow a path over another router or it can divert them towards itself.
  - *Detour attack.* An attacker forces its neighbors to route all their network traffic over it.

Related Work. In [8, 11, 13, 5, 4], the authors have developed techniques to counter some of the above-mentioned security threats in classical routing algorithms. They utilized standard cryptography techniques i.e. digital signatures or Hashed Message Authentication Code (HMAC) to avert fabrication and tampering attacks. In these techniques, a router verifies that the originator of a control message is the node that is indicated in the header. In [11], the authors have secured distance vector routing protocols by incorporating the information about a node and its predecessor node in the control packet. Sequence numbers are used to identify an old or obsolete control packet. However, none of these approaches try to analyze and counter the security threats related to the specific features of Nature inspired routing algorithms with the exception of the preliminary work of Zhong and Evans [16]. They studied the anomalous behavior of *AntNet* [1] under three types of attacks: fabrication, dropping and tampering. Their experiments clearly demonstrate that the malicious nodes can disrupt the normal routing behavior of *AntNet* by launching these attacks.

3 BeeHive Algorithm

This algorithm was proposed by Wedde, Farooq and Zhang in [15]. The algorithm is inspired by the communication language of honey bees. Each node periodically sends a *bee agent* by broadcasting the replicas of it to each neighbor. The replicas explore the network using priority queues and they use an estimation model to estimate the propagation and queuing delay from a node, where they are received, to their launching node. Once the replicas of the same agent arrive at a node via different neighbors of the node, they exchange routing information to model the network state at this node. Through this exchange of information by the replicas at a node, the node is able to maintain a quality metric for reaching destinations via its neighbors. The algorithm utilizes just forward moving agents and no statistical parameters are stored in the routing tables. In *BeeHive* a network is divided into *Foraging Regions* and *Foraging Zones.* Each node belongs to only one *Foraging Region.* Each *Foraging Region* has a representative node. A *Foraging Zone* of a node consists of all the nodes from which a replica of an agent could reach this node in 7 hops. This approach significantly reduces the size of the routing table because each node maintains detailed routing information only about reaching the nodes within its *Foraging Zone* and for reaching the representative nodes of the *Foraging Regions.* In this way, a data packet, whose destination is beyond the *Foraging Zone* of a node, is forwarded in the direction of the representative node of the *Foraging Region* containing the destination node. The next hop for a data packet at a node is selected in a probabilistic fashion depending upon the goodness of each neighbor for reaching the destination. *BeeHive* is also fault-tolerant to crashing of routers. The interested reader will find more details in [15].

4 BeeHiveGuard

In our work, the basic motivation is to analyze the security threats of deploying Nature inspired routing protocols and then to design and develop a comprehensive security framework that can counter these threats. As a first step, we take the *BeeHive* algorithm, introduced in the previous section, and extend it with our security model that can counter these threats. We applied standard RSA algorithm [10] for doing cryptography functions like encryption and decryption. We name the new algorithm as *BeeHiveGuard* and discuss its relevant security features. We assume that a secure key distribution infrastructure exists in the network.

Agent integrity. The purpose of this extension is that the management information related to a *bee agent* can not be modified/impersonated by an intermediate router. The values of relevant information fields of a *bee agent* that must be protected are: its identifier and identifier of its replicas, its source address, its time to live timer (TTL) and the address of its *Foraging Region*. The source node signs these fields with its private key and puts the corresponding signature *sig1* in the *bee agent*. If a traitorous router tries to change these fields or impersonate someone else then other nodes can easily detect and discard the corresponding bogus *bee agents*.

Routing information integrity. The purpose of this extension is to secure the routing information i.e. the propagation delay or the queuing delay of a *bee agent*. The delays are used to estimate the quality of a visited path. The routers calculate the delay values and then modify them accordingly in the *bee agents*, therefore, it becomes a challenging task to differentiate a valid modification from a fake one. We can do it if we assume that *no two subsequent routers on a route fake their routing information*. The basic idea is that a *bee agent* carries the signed routing information of a node and its predecessor node. *sig2* is the signature obtained by signing the queuing and propagation delays of a visited node and *sig3* is the signature for its predecessor node. Figure 1 shows how digital signatures are used to secure the routing information.

Node 1 launches two replicas of its *bee agent* towards Node 0 and Node 2 respectively. Node 1 has no predecessor, therefore, *sig2* represents the signed delays of the Node 1 and *sig3* is obtained by signing a 0 value for both delays. Once the replica arrives at Node 2 then *sig3* is set to *sig2* and the delays of this node are signed in *sign2*. This process continues until the replica reaches Node

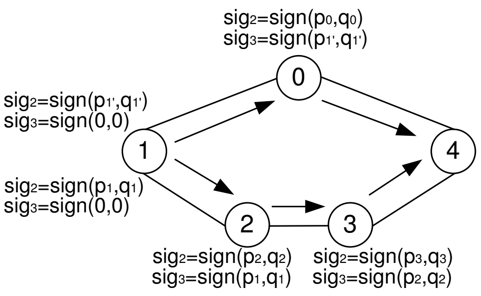


Fig. 1. Securing routing information in BeeHiveGuard

4. Then Node 4 estimates its delays to Node 1 by adding its delay values with the ones of Node 3 and Node 2. As a result, Node 3 can only manipulate its own delays but not the cumulative delays from Node 1 to Node 3. Moreover, a node also compares the delay values in *sig2* with the ones in *sig3*. If the delay values in *sig2* are lesser or equal to the ones in *sig3* then the predecessor node has provided fake delay values. As a result, *sig2* value at this node is calculated with the help of the delay values in *sig3* and the *bee agent* continues its exploration. Since a node utilizes the information of its predecessor node and the predecessor node of its predecessor node, therefore, a predecessor node can not significantly influence the routing behavior by faking its own routing information above.

5 Experiments

We designed a series of experiments, which simulate different types of threats in the networks. The results of the experiments clearly demonstrate that *BeeHive* algorithm is susceptible to a number of such attacks. We utilized a standard cryptography library, OpenSSL [2], to implement our security model in the *BeeHiveGuard* algorithm. The library supports relevant cryptography techniques like digital signatures, symmetric and asymmetric cryptography, and cryptography hash functions. A profiling framework, which measures the processing complexity of a function in cycles, is incorporated in the performance evaluation framework presented in [14]. The empirical validation of our security model is necessary because it is not a trivial task to formally model the emergent behavior of Nature inspired routing protocols. *BeeHiveGuard* is realized in OMNeT++ simulator [12]. The experiments were conducted on Fujitsu Siemens PC, which has a Pentium 4 3.0 GHz processor and 1 Gigabyte of RAM. The reported values are an average of the values obtained from ten independent runs.

Tampering control messages. We simulated in topology Net1 a malicious node, which alters the queuing delay and propagation delay fields of the *bee agents* passing through it. In Net1, a traffic session is started between Node 4 and Node 1. Node 3 modifies the queuing and propagation delays of the *bee*

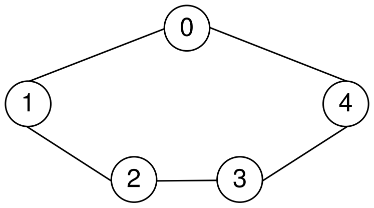


Fig. 2. Net1: Node 3 is tampering the queuing and the propagation delay

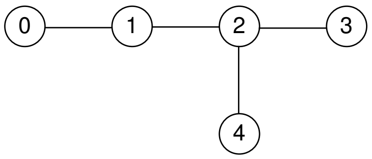


Fig. 3. Net2: Impersonating, Detour and Flooding attacks launched by Node 4

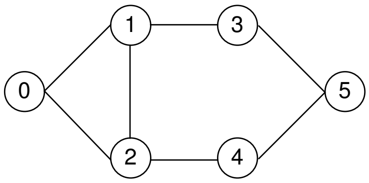


Fig. 4. Net3: Rushing and dropping Attacks

agents launched by Node 1. As a result, it artificially increases the quality of the path 4-3-2-1 as compared to 4-0-1.

Impersonating, Detour and Flooding attacks. In topology Net2, we created a scenario which can simulate impersonating, detour and flooding attacks. A traffic session is started between Node 3 and Node 0. In a normal mode, data packets take the path 3-2-1-0. However, Node 4 launches the three attacks by injecting a large number of *bee agents*, which have Node 0 as their source node instead of Node 4. In this way, data packets also take the path 3-2-4-2-1-0.

Rushing and Dropping attacks. In topology Net3, we created a traffic session between Node 5 and Node 0. In this scenario, Node 1 retransmits the *bee agents* from Node 0 by increasing their *agent id* and changing their *replica id* to that of the replicas of the path 0-2-*. As a result, Node 2 always drops the new *bee agents*, which arrive directly from Node 0 because their *agent id* is smaller than the ones which arrive over the path 0-1-2. Consequently, if the route 5-4-2-0 initially had a poor quality then its quality could never improve because Node 5 would always get the bogus *bee agents* through the path 0-1-2-4-5.

6 Results

Tampering control messages. In Figure 5, one can see that in a normal mode the path 4-0-1 is rated higher than the path 4-3-2-1 because of its smaller delays. As a result, more data packets are routed on the path 4-0-1 as compared with the path 4-3-2-1. However, the situation is drastically changed once Node 3 launches its attack at 300 seconds by tampering the information in the *bee agents* (see Figure 6). The impact of the attack is significantly reduced in *BeeHiveGuard*

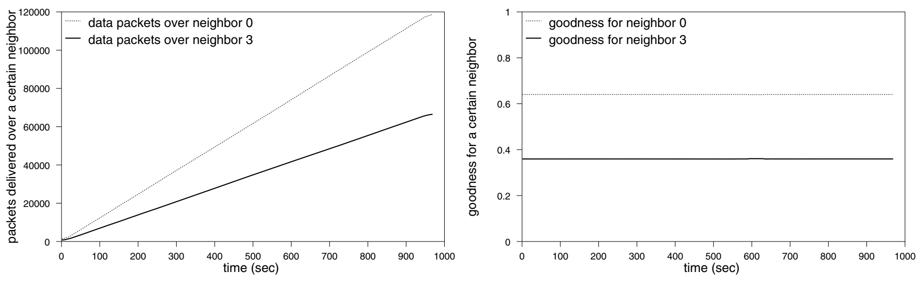


Fig. 5. Net 1: Normal mode (measurements are made at Node 4)

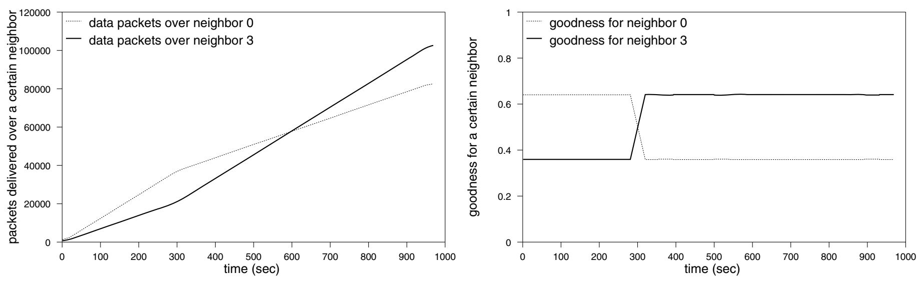


Fig. 6. Net 1: Under attack (measurements are made at Node 4)

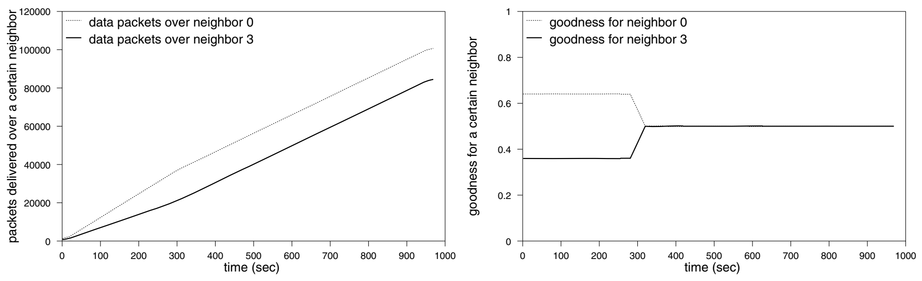


Fig. 7. Net 1: Secure mode (measurements are made at Node 4)

(see Figure 7) because Node 3 can now just manipulate its own queuing and propagation delays. Remember in *BeeHive* it can manipulate the delays of the complete path 3-2-1.

Impersonating, Detour and Flooding attacks. One can see in Figure 8 that in a normal mode all data packets are routed on the path 3-2-1-0. Node 4 launches its attacks by transmitting bogus *bee agents* at 300 seconds. As a result, it has successfully detoured a significantly large number of data packets towards itself (see Figure 9). Please remember that the left subfigure of Figure 9 shows

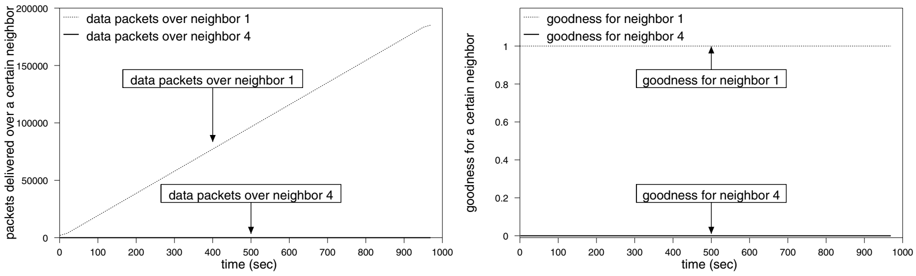


Fig. 8. Net 2: Normal mode (measurements are made at Node 2)

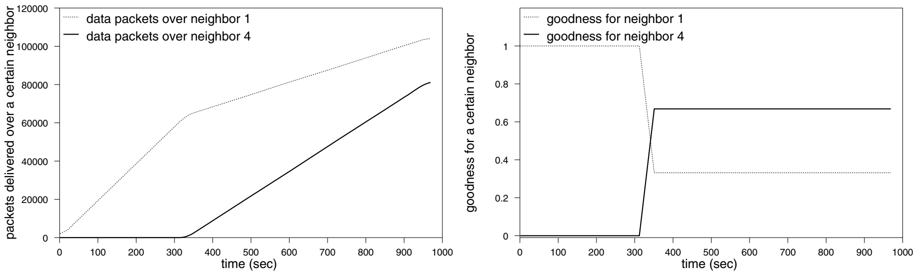


Fig. 9. Net 2: Under attack (measurements are made at Node 2)

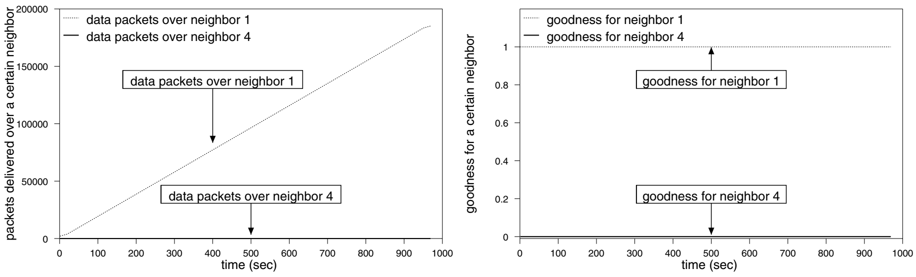


Fig. 10. Net 2: Secure mode (measurements are made at Node 2)

the number of packets that followed either the path 3-2-1 or 3-2-4. Consequently, the number of data packets that followed the path 3-2-4-2-1 is not counted for neighbor 1. Figure 10 shows that in *BeeHiveGuard* Node 4 is not able to influence the routing decisions by propagating its bogus *bee agents*.

Rushing and Dropping attacks. It is obvious from Figure 11 that in a normal mode *BeeHive* is able to achieve excellent load balancing in the steady state by distributing the packets on the paths 5-3-1-* and 5-4-2-*. However, once Node

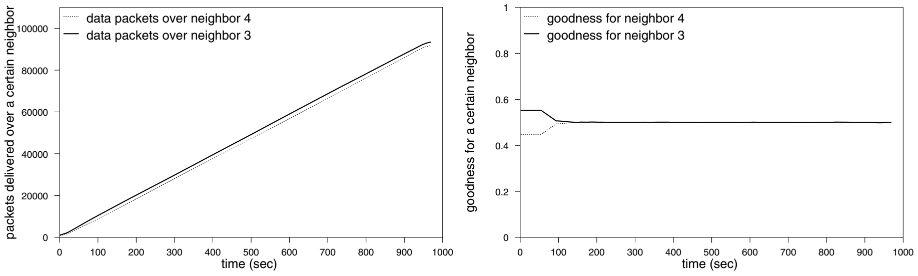


Fig. 11. Net 3: Normal mode (measurements are made at Node 5)

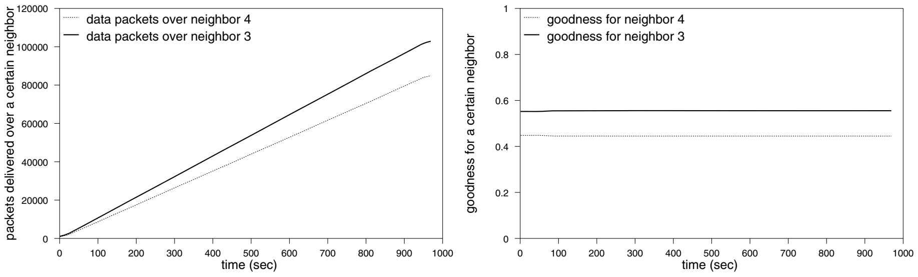


Fig. 12. Net 3: Under attack (measurements are made at Node 5)

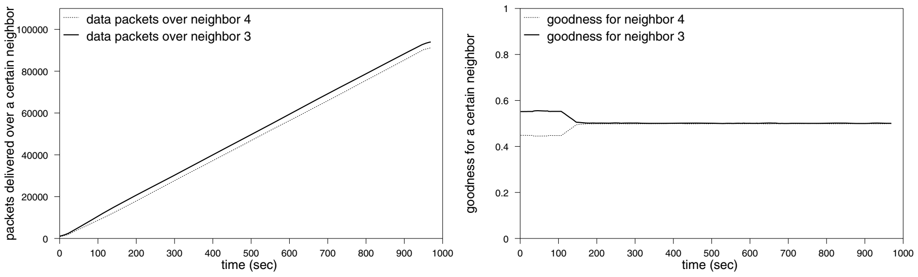


Fig. 13. Net 3: Secure mode (measurements are made at Node 5)

1 launches its attack by retransmitting the *bee agents* of Node 0 with modified agent and replica ids then the quality of the paths 5-4-2-* does not improve. As a result, Node 1 is able to divert the network traffic towards itself through the path 5-3-1. However, the impact in this attack is less significant as compared with the previous ones (see Figure 12). Figure 13 shows that *BeeHiveGuard* has successfully countered even these attacks because Node 1 could not modify the *bee agents* launched by Node 0.

7 Costs of BeeHiveGuard

We have collected relevant cost and performance values in Table 1. A_a is the average number of processor cycles required to process a *bee agent*, R_o models the bandwidth consumed by the *bee agents*, S_o models the additional bandwidth consumed by data packets if they do not follow the shortest hop path, T_{av} is the average throughput (Mbits/sec) and t_d is the average delay (in milliseconds). It is easy to conclude from Table 1 that *BeeHiveGuard* is able to counter the threats and provide the same throughput and packet delay (the two important performance values) than that of *BeeHive* in a normal mode. However, the security is achieved at a considerable processing and communication costs. In Net1, the average processing complexity and the communication overhead of *bee agents* increased to 98600% and 750% respecitvely as compared to the *bee agents* in *BeeHive*. A similar tendency is observed in case of Net2 and Net3 (see Table 1). The significant increase in processing cost is due to complex mathematical operations that are performed in classical cryptography. The *bee agents* now carry additional fields like digital signatures and this results in a substantial increase in their size. As a result, the communication overhead is also significantly increased.

Table 1. Costs of BeeHiveGuard

Topology	Algorithm	A_a	R_o	S_o	T_{av}	t_d
Net1	BeeHive	16699	0.022	0.473	0.79	0.004
Net1	BeeHive - Attack (1)	16547	0.022	0.73	0.79	0.004
Net1	BeeHiveGuard (2)	16341381	0.187	0.587	0.79	0.004
Difference (1) and (2) in %		98600	750	19.5	0	0
Net2	BeeHive	13924	0.018	0	0.79	0.004
Net2	BeeHive - Attack (3)	14703	0.057	1.443	0.79	0.008
Net2	BeeHiveGuard (4)	23603742	0.133	0	0.79	0.004
Difference (3) and (4) in %		160400	133.3	100	0	50
Net3	BeeHive	21219	0.029	0.133	0.79	0.005
Net3	BeeHive - Attack (5)	25353	0.017	0.29	0.788	0.007
Net3	BeeHiveGuard (6)	13247990	0.254	0.132	0.79	0.005
Difference (5) and (6) in %		52100	1390	54.4	0.2	28.5

8 Conclusion and Future Work

We analyzed the security threats of a Nature inspired routing protocol, *BeeHive*. However, the analysis can be easily extended to any Nature inspired routing protocol. The results of our extensive experiments reveal that the algorithm is susceptible to a number of Byzantine attacks like tampering, impersonating, detour, flooding, rushing and dropping that a malicious node can launch in the network. The results of the experiments confirm that *BeeHiveGuard* can successfully counter all of these attacks. *BeeHiveGuard* utilizes standard cryptography techniques for secure routing.

We extended an existing performance evaluation framework to measure the processing complexity and communication costs of our security model. Our results indicate that the standard cryptography techniques are not feasible for Nature inspired routing algorithms because the average agent complexity and communication costs of a *bee agent* in *BeeHiveGuard* lies in the range from 52100% to 160400% and from 133% to 1390% respectively, as compared to *BeeHive*. This overhead of providing security will hinder the normal packet switching task of a real world router. The important conclusion of our current work is: *if Nature inspired routing protocols are to be deployed in real world networks then a novel security architecture is to be developed whose processing and communication overheads are significantly smaller as compared with the existing cryptography techniques*. We believe that Artificial Immune Systems (AIS) [3] is one such alternative paradigm. Our objective is to design and develop a comprehensive and scalable security framework that does not use complex cryptography techniques but provides the same security level. This will be the subject of our forthcoming publications.

References

1. G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communication networks. *Journal of Artificial Intelligence Research*, 9:317–365, December 1998.
2. M. Cox, R. Engelschall, S. Henson, and B. Laurie. The openssl project. http://www.openssl.org.
3. Dipankar Dasgupta, editor. *Artificial Immune Systems and their Applications*. Springer-Verlag, 1998.
4. R. Hauser, T. Przygienda, and G. Tsudik. Reducing the cost of security in link-state routing. In *SNDSS '97: Proceedings of the 1997 Symposium on Network and Distributed System Security*, page 93, Washington, DC, USA, 1997. IEEE Computer Society.
5. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Efficient security mechanisms for routing protocolsa. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. The Internet Society, 2003.
6. Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1-2):21–38, 2005.

7. S. Liang, A. N. Zincir-Heywood, and M. I. Heywood. Intelligent packets for dynamic network routing using distributed genetic algorithm. In *Proceedings of Genetic and Evolutionary Computation Conference*. GECCO, July 2002.
8. S. L. Murphy and M. R. Badger. Digital signature protection of the ospf routing protocol. In *SNDSS '96: Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 93, Washington, DC, USA, 1996. IEEE Computer Society.
9. R. Perlman. Network layer protocols with byzantine robustness. phd thesis. Technical report, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technolog, 1998.
10. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
11. Bradley R. Smith, Shree Murthy, and J. J. Garcia-Luna-Aceves. Securing distance-vector routing protocols. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 1997, San Diego, California, USA*. IEEE Computer Society, 1997.
12. A. Varga. OMNeT++: Discrete event simulation system: User manual. http://www.omnetpp.org.
13. B. Vetter, F. Wang, and S. F. Wu. An experimental study of insider attacks for ospf routing protocol. In *ICNP '97: Proceedings of the 1997 International Conference on Network Protocols (ICNP '97)*, page 293, Washington, DC, USA, 1997. IEEE Computer Society.
14. H. F. Wedde and M. Farooq. A performance evaluation framework for nature inspired routing algorithms. In *Applications of Evolutionary Computing, LNCS 3449*, pages 136–146. Springer Verlag, March 2005.
15. H. F. Wedde, M. Farooq, and Y. Zhang. BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In *Ant Colony Optimization and Swarm Intelligence, LNCS 3172*, pages 83–94. Springer Verlag, Sept 2004.
16. W. Zhong and D. Evans. When ants attack: Security issues for stigmergic systems. Technical report, Department of Computer Science, University of Virginina, CS-2002-3, 2002.

Optimal Broadcasting in Metropolitan MANETs Using Multiobjective Scatter Search*

F. Luna[1], A.J. Nebro[1], B. Dorronsoro[1], E. Alba[1], P. Bouvry[2], and L. Hogie[2]

[1] Department of Computer Science, University of Málaga, Spain
{flv, antonio, dorronsoro, eat}@lcc.uma.es
[2] Faculty of Sciences, Technology and Communications, University of Luxembourg
{pascal.bouvry, luc.hogie}@uni.lu

Abstract. Mobile Ad-hoc Networks (MANETs) are composed of a set of communicating devices which are able to spontaneously interconnect without any pre-existing infrastructure. In such scenario, broadcasting becomes an operation of capital importance for the own existence and operation of the network. Optimizing a broadcasting strategy in MANETs is a multiobjective problem accounting for three goals: reaching as many stations as possible, minimizing the network utilization, and reducing the makespan. In this paper, we face this multiobjective problem with a state-of-the-art multiobjective scatter search algorithm called AbSS (Archive-based Scatter Search) that computes a Pareto front of solutions to empower a human designer with the ability of choosing the preferred configuration for the network. Results are compared against those obtained with the previous proposal used for solving the problem, a cellular multiobjective genetic algorithm (cMOGA). We conclude that AbSS outperforms cMOGA with respect to three different metrics.

1 Introduction

Mobile Ad-hoc Networks (MANETs) are fluctuating networks populated by a set of communicating devices called *stations* (they are also called *terminals*) which can spontaneously interconnect each other without a pre-existing infrastructure. This means that no carrier is present in such networks as it is usual in many other types of communication networks. Stations in MANETs are usually laptops, PDAs, or mobile phones, equipped with network cards featuring wireless technologies such as Bluetooth and/or IEEE802.11 (WiFi). In this scenario, a) stations communicate within a limited range, and b) stations can move while communicating. A consequence of mobility is that the topology of such networks may change quickly and in unpredictable ways. This dynamical behavior constitutes one of the main obstacles for performing efficient communications on such networks.

Broadcasting is a common operation at the application level and is also widely used for solving many network layer problems being, for example, the basis

* This work has been partially funded by the Ministry of Science and Technology and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project).

mechanism for many routing protocols. In a given MANET, due to host mobility, broadcasting is expected to be performed very frequently (e.g., for paging a particular host, sending an alarm signal, and/or finding a route to a given target terminal). Broadcasting may also serve as a last resort to provide multicast services in networks with such rapidly changing topologies and stems for the organization of terminals in groups. Hence, having a well-tuned broadcasting strategy will result in a major impact in network performance.

In this paper we are considering the problem of broadcasting on a particular sub-class of MANETs called *Metropolitan* MANETs, which cover from shopping malls to metropolitan areas. Instead of providing a generic protocol performing well on average situations, our proposal consists of optimally tuning the broadcasting service for a set of networks and for a particular category of broadcast messages. Optimizing a broadcasting strategy is a multiobjective problem where multiple functions have to be satisfied at the same time: maximizing the number of stations reached, minimizing the network use, and minimizing the makespan are three examples of the potential objectives. In this work, the broadcasting strategy considered for optimization is DFCN [1], and the target networks are metropolitan MANETs. Since manipulating such networks is difficult, we must rely on software simulators for evaluating the scenarios from the designer point-of-view.

Contrary to single objective optimization, multiobjective optimization is not restricted to find a unique solution of a given multiobjective problem, but a set of solutions known as the *Pareto optimal set.* For instance, taking as an example the problem we are dealing with, one solution can represent the best result concerning the number of reached stations, while another solution could be the best one concerning the makespan. These solutions are said to be *nondominated.* The result provided by a multiobjective optimization algorithm is then a set of nondominated solutions (the *Pareto optima*) which are collectively known as the *Pareto* front when plotted in the objective space. The mission of the decision maker is to choose the most adequate solution from the Pareto front.

This multiobjective problem of broadcasting in MANETs, which has been previously addressed with a cellular genetic algorithm (cMOGA) in [2], is now tackled with a state-of-the-art multiobjective scatter search algorithm called AbSS (Archive-based Scatter Search) [3]. Scatter search [4, 5, 6] has been successfully applied to a wide variety of optimization problems [5], but it has not been extended to deal with MOPs until recently [3, 7, 8, 9]. This metaheuristic technique starts from an initial set of diverse solutions from which a subset, known as the *reference set* (*RefSet*), is built by including both high quality solutions and highly diverse solutions. Then, an iterative procedure systematically combines the solutions in RefSet somehow for generating new (hopefully better) solutions that may be used for updating the reference set and even the initial population. After that, an iterative procedure is used to locate an optimal solution.

The contributions of this work are summarized in the following. Firstly, we solve the broadcasting problem on MANETs using a multiobjective scatter search, and compare the results with those obtained with cMOGA. Secondly, we are dealing in this work with a more realistic problem than the one faced in [2] because we are using an interesting real world scenario (a shopping mall) never tackled before.

The rest of the paper is structured as follows. In the next section, we detail the multiobjective problem of broadcasting in MANETs. Section 3 includes the description of the multiobjective scatter search algorithm. Metrics, parameterization, and results are presented in Sect. 4. Finally, conclusions and lines of future work are given in Sect. 5.

2 Problem Definition

The problem we study in this paper consists of, given an input MANET, determining the most adequate parameters for a broadcasting strategy in it. We first describe in Sect. 2.1 the target networks we have used. Section 2.2 is devoted to the presentation of DFCN, the broadcasting strategy to be tuned. Finally, the MOP we define for this work is presented in Sect. 2.3.

2.1 Metropolitan Mobile Ad-Hoc Networks

Metropolitan mobile ad-hoc networks are MANETs with some particular properties. Firstly, they have one or more areas where the node density is higher than the average. These points are called VHS, standing for *Virtual Hot Spots*, that can be statistically detected. A VHS may be, for example, a shopping center, an airport, or an office. Secondly, virtual hot spots do not remain active full time, i.e., they can appear and disappear from the network (e.g., supermarkets are open, roughly, from 9 a.m. to 9 p.m., and outside this period of time, the node density within the corresponding area is close to zero).

To deal with such kind of networks, we have to rely on software simulators. In this work we have used *Madhoc* [1], a metropolitan MANET simulator. It aims at providing a tool for simulating different level services based on different technologies on MANETs for different environments, ranging from open areas to metropolitan ones. In order to make more realistic the simulations, Madhoc has been endowed with an observation window such that only the devices located inside this window are taken into account for measurements. Hence, we allow the existence of a changing number of devices in the network as it happens in real MANETs. This recent feature of Madhoc is displayed in Fig. 1, where both an example of a metropolitan MANET (a) and the effects of introducing an observation window on it (b) are shown. We highlight as well a typical action of devices going in and leaving the window in the right part of the figure. In all the tests in this work, this observation window is 70% of the total simulation area. The main parameters of Madhoc used for defining the network characteristics are the following:

[1] http://www-lih.univ-lehavre.fr/~hogie/madhoc/

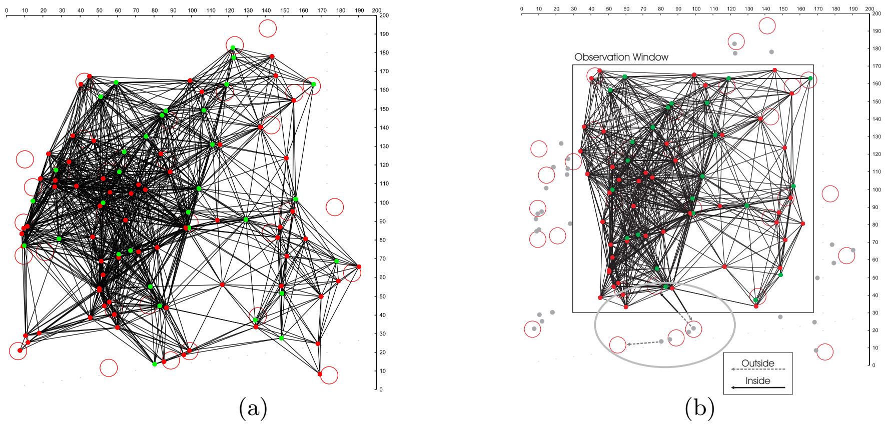


Fig. 1. (a) Metropolitan MANET, and (b) the effect of the observation window

***size*:** defines the network simulation area in terms of square meters.
***density*:** is the average density of nodes per square kilometer (i.e., the number of devices per square kilometer).
***environment*:** determines the mobility model for the stations and the radio wave propagation model. That is, this feature defines how the stations are moving as well as the area within which they are moving (open areas, buildings, streets, etc.), thus determining how radio waves are propagated.

2.2 Delayed Flooding with Cumulative Neighborhood

Broadcasting strategies in MANETs can be classified into four categories: simple flooding, probability-based methods, area-based methods, and neighbor-knowledge-based methods (a survey can be found in [10]). This categorization is based on the way that protocols select re-broadcasting stations.

Broadcasting protocols can also be classified depending on whether they deal with mobility or not. The vast majority of present protocols do not consider any active management of station mobility. The Delayed Flooding with Cumulative Neighborhood (DFCN) protocol belongs to the neighbor-knowledge-based class, and it features an active management of station mobility so it is able to make new broadcasting decisions on new neighbor discovery. For being able to run the DFCN protocol, the following assumptions must be met:

– Like many other neighbor-knowledge-based broadcasting protocols (FWSP, SBA, etc.), DFCN requires the knowledge of 1-hop neighborhood, which can be obtained by using "hello" packets at a lower network layer. The set of neighbors of station s is named $N(s)$.
– Each message m carries —embedded in its header— the set of IDs of the 1-hop neighbors of its most recent sender.
– Each station maintains local information about all the messages received. Each instance of this local information consists of the following items:

  - the ID of the message received;
  - the set of IDs of the stations that are known to have received the message;
  - the decision of whether the message should be forwarded or not.
- DFCN requires the use of a random delay before possibly re-emitting a broadcast message m. This delay, called Random Assessment Delay (RAD), is intended to preventing collisions. More precisely, when a station s emits a message m, all the stations in $N(s)$ receive it at the same time. It is then likely that all of them forward m simultaneously, and this simultaneity entails network collisions. The RAD aims at randomly delaying the retransmission of m. As every station in $N(s)$ waits for the expiration of a different RAD before forwarding m, the risk of collisions is hugely reduced.

DFCN is an event driven algorithm which can be divided into three main parts: the two first ones deal with the station handling of outcoming events, which are (1) new message reception and (2) detection of a new neighbor. The third part (3) consists of the decision making of the station for emission as a follow-up of one of the two previous events. The behavior resulting from message reception is referred to as *reactive* behavior; when a new neighbor is discovered, the behavior is referred as *proactive* behavior.

Let s_1 and s_2 be two stations in the neighborhood of one another. When s_1 sends a packet to s_2, it attaches the set $N(s_1)$ to the packet. At reception, s_2 hence knows that each station in $N(s_1)$ has received the packet. The set of stations which have *potentially* not yet received the packet is then $N(s_2) - N(s_1)$. If s_2 re-emits the packet, the *effective* number of stations newly reached is maximized by the heuristic function: $h(s_2,\ s_1) = |N(s_2) - N(s_1)|$.

In order to minimize the network overload caused by a possible packet re-emission, this re-emission occurs only if the number of newly reached stations is greater than a given threshold. This threshold is a function of the number of stations in the neighborhood (the local network density) of the recipient station s_2. It is written $threshold(|N(s)|)$. The decision made by s_2 to re-emit the packet received from s_1 is defined by the boolean function:

$$\text{Re-emit}\,(s_2,\ s_1) = \begin{cases} \text{true} & \text{h}\,(s_2,\ s_1)\ \geq\ \text{threshold}\,(|N\,(s_2)|) \\ \text{false} & \text{otherwise}\ . \end{cases} \quad (1)$$

If the threshold is exceeded, the recipient station s_2 becomes an emitter after a random delay defined by RAD. The threshold function, which allows DFCN to facilitate the message re-broadcasting when the connectivity is low, depends on the size of the neighborhood n, as given by:

$$\text{threshold}(n) = \begin{cases} 1 & n \leq\ \text{safeDensity} \\ \text{minGain} * n & \text{otherwise}\ . \end{cases} \quad (2)$$

where $safeDensity$ is the maximum safe density below which DFCN always rebroadcasts and *minGain* is the minimum gain for rebroadcasting, i.e., the ratio between the number of neighbors which have not received the message and the total number of neighbors.

Each time a station s gets a new neighbor, the RAD for all messages is set to zero and, therefore, messages are immediately candidate to emission. If $N(s)$ is greater than a given threshold, which we have called *proD*, this behavior is disabled, so no action is undertaken on new neighbor discovery. *proD* is used for avoiding massive packet rebroadcasting when a new station appears in highly dense areas, that is, avoiding network congestions on the proactive behavior.

2.3 MOP Definition: DFCNT

From the description of the previous section, the following DFCN parameters are to be tuned:

minGain is the minimum gain for rebroadcasting. This is the most important parameter for tuning DFCN, since minimizing the bandwidth should be highly dependent on the network density. It ranges from 0.0 to 1.0.

[lowerBoundRAD,upperBoundRAD] defines the RAD value (random delay for rebroadcasting in milliseconds). Both parameters take values in the interval $[0.0, 10.0]$ milliseconds.

proD is the maximal density ($proD \in [0, 100]$) for which it is still needed using proactive behavior (i.e., reacting on new neighbors) for complementing the reactive behavior.

safeDensity defines a maximum safe density of the threshold which ranges from 0 to 100 devices.

These parameters, i.e., a DFCN configuration, characterize the search space. Here, the objectives to be optimized are: minimizing the makespan (in seconds), maximizing the network coverage (percentage of devices having received the broadcasting message), and minimizing the bandwidth used (in number of transmissions). Thus, we have defined a triple objective MOP, which has been called DFCNT (standing for DFCN Tuning). For obtaining the values of these objective functions we have used Madhoc because it implements the DFCN broadcasting protocol. Then, our goal is to obtain the Pareto front of DFCNT (and the corresponding DFCN configurations) in terms of these three objectives.

3 Multiobjective Scatter Search

In this section, we first give a brief overview of the scatter search technique and, second, we describe the modifications on this standard scatter search for dealing with MOPs to explain our proposed AbSS.

3.1 Scatter Search

Most implementations of scatter search use the template proposed by Glover in [4]. As depicted in Fig. 2, this metaheuristic consists of five methods: diversification generation, improvement, reference set update, subset generation, and solution combination.

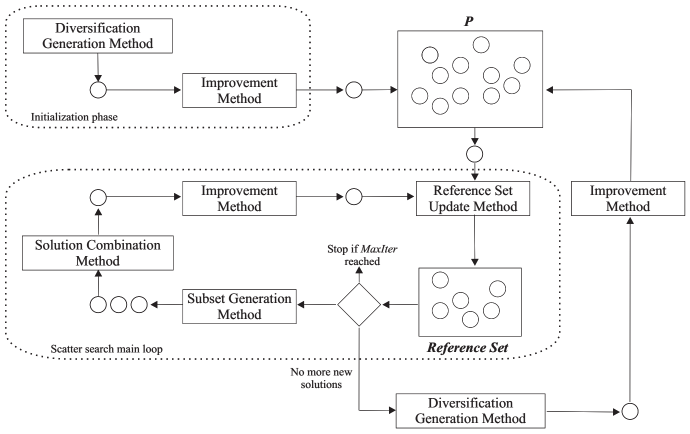


Fig. 2. Outline of the standard scatter search algorithm

The scatter search technique starts by creating an initial set of diverse individuals in the initialization phase. This phase consists of iteratively generating new solutions by invoking the diversification generation method; each solution is passed to the improvement method, which usually applies a local search procedure in an iterative manner, and the resulting individual is included into the initial set P. After the initial phase, the scatter search main loop starts.

The main loop begins building the reference set from the initial set by invoking the reference set update method. The reference set is a collection of both high quality solutions and diverse solutions that are used for generating new individuals. Solutions in this set are systematically grouped into subsets of two or more individuals by means of the subset generation method. In the next step, solutions in each subset are combined to create a new individual, according to the solution combination method. Then, the improvement method is applied to every new individual. The final step consists of deciding whether the resulting solution is inserted into the reference set or not. This loop is executed until a termination condition is met (for example, a given number of iterations has been performed, or the subset generation method does not produce new subsets).

Optionally, there is a re-start process invoked when the subset generation method does not produce new subsets of solutions. The idea is to obtain a new initial set, which will now include the current individuals in the reference set. The rest of individuals is generated by using the diversification generation and improvement methods, as in the initial phase.

3.2 AbSS

AbSS (Archive-based Scatter Search) [3] is based on the aforementioned scatter search template and its application to solve bounded continuous single objective optimization problems [6]. It uses an external archive for storing nondominated solutions and combines ideas of three state-of-the-art evolutionary algorithms for solving MOPs. In concrete, the archive management follows the scheme of

PAES [11], but using the crowding distance of NSGA-II [12] as a niching measure instead of the PAES adaptive grid; additionally, the density estimation found in SPEA2 [13] is adopted for selecting the solutions from the initial set that will build the reference set. Once described the overall view of the technique, we now detail the five methods to engineer AbSS:

- **Diversification Generation Method:** Its goal is to generate an initial set P of diverse solutions. The method consists of dividing, for every new solution, the range of each variable into a number of subranges of equal size; then, each solution is created in two steps. Firstly, a subrange is randomly chosen, with the probability of selecting a subrange being inversely proportional to its frequency count (the number of times the subrange has been previously selected); secondly, a value is uniformly randomly generated within the selected range.
- **Improvement Method:** It is a local search method based on a mutation operator (Polynomial mutation [14]) and a Pareto dominance test. It operates by iteratively mutating an individual with the aim of improving it. Since we are dealing with MOPs, it may occur that the newly generated individual and the current one are nondominated each other (Pareto dominance test). In this case, the original individual is inserted into the external archive and the mutated individual becomes the new current one.
- **Reference Set Update Method:** A similar issue rises when building the RefSet in this method, i.e., how to pick up the best among a set of nondominated solutions. RefSet is composed of two subsets, $RefSet_1$ and $RefSet_2$ so that the first one contains the best quality solutions in the initial set of solutions, while the second subset should be filled with solutions promoting diversity. While $RefSet_2$ is constructed by choosing those individuals whose minimum Euclidean distance to the reference set is the highest, $RefSet_1$ is built by using the concepts of strength raw fitness and a density estimation of SPEA2 [13] when choosing the best individuals.
- **Subset Generation Method:** It generates all pairwise combinations of solutions in RefSet_1 and, separately, in RefSet_2.
- **Solution Combination Method:** The simulated binary crossover (SBX) [14] is used for combining solutions in AbSS.

4 Experiments

This section is devoted to presenting the experiments performed for this work. We first describe the metrics used for measuring the performance of the resulting Pareto fronts. Next, the parameterization of AbSS and Madhoc is detailed. Finally, we show the results for DFCNT and compare them against cMOGA [2].

4.1 Metrics

We have used three metrics for assessing the performance of both AbSS and cMOGA: the number of Pareto optima that the optimizers are able to find, Set

Coverage [15] which allows two algorithms to be compared in terms of Pareto dominance, and Hypervolume [16] which measures both convergence and diversity at the same time in the resulting Pareto fronts. They are defined as:

- **Number of Pareto optima:** Given that DFCNT is a difficult problem, finding a high number of nondominated solutions could be itself a hard challenge for any multiobjective optimizer. In this sense, the number of Pareto optima can be considered as a measure of the ability of the algorithm for exploring difficult search spaces defined by hard MOPs like DFCNT.
- **Set Coverage:** The set coverage metric $\mathcal{C}(A,B)$ calculates the proportion of solutions in B which are dominated by solutions of A: $\mathcal{C}(A,B) = \frac{|\{b \in B \mid \exists a \in A : a \preceq b\}|}{|B|}$.

  A metric value $\mathcal{C}(A,B) = 1$ means that all members of B are dominated by A, whereas $\mathcal{C}(A,B) = 0$ means that no member of B is dominated by A. This way, the larger the $\mathcal{C}(A,B)$, the better the Pareto front A with respect to B. Since the dominance operator is not symmetric, $\mathcal{C}(A,B)$ is not necessarily equal to $1 - \mathcal{C}(B,A)$, and both $\mathcal{C}(A,B)$ and $\mathcal{C}(B,A)$ have to be computed for understanding how many solutions of A are covered by B and vice versa.
- **Hypervolume:** This metric calculates the volume (in the objective space) covered by members of a nondominated set of solutions Q. Let v_i be the volume enclosed by solution $i \in Q$. Then, a union of all hypercubes is found and its hypervolume (HV) is calculated: $HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right)$.

  Algorithms with larger values of HV are desirable. Since this metric is not free from arbitrary scaling of objectives, we have evaluated the metric by using normalized objective function values.

4.2 Parameterization

As we stated in Sect. 2.1, the behavior of Madhoc has been defined based on three parameters mainly: the size of the simulation area, the density of mobile stations, and the type of environment. For our experiments, we have used a simulation area of 40,000 square meters, a density of 2,000 stations per square kilometer, and, from the available environments of Madhoc, the mall environment has been used. This environment is intended to model a commercial shopping center, in which stores are usually located together one each other in corridors. People go from one store to another by these corridors, occasionally stopping for looking at some shopwindows. Both the mobility of devices and their signal propagation are restricted due to the walls of the building. A metropolitan MANET with such a configuration has been shown in Fig. 1. Due to the stochastic nature of Madhoc, five simulations (i.e., five different network instances) per function evaluation have been performed so that the fitness values of the functions are computed as the average resulting values of these five different network instances.

The configuration used for cMOGA is the same as that used in [2]: a population of 100 individuals arranged in a 10×10 square toroidal grid, the neighborhood is NEWS, binary tournament selection, simulated binary crossover (SBX)

Table 1. Performance metrics for AbSS and cMOGA when solving DFCNT

Metric	AbSS		cMOGA		t-test
	average	std	average	std	
Number of Pareto Optima	98.7586	2.8119	98.1053	2.9000	–
Set Coverage	0.9865	0.0103	0.9793	0.0076	+
Hypervolume	0.8989	0.0695	0.8199	0.0854	+

with $p_c = 1.0$, polynomial mutation ($p_m = 1.0/L$, $L =$ individual length), archive size of 100 individuals, and the adaptive grid of PAES [11] has been used as crowding method (see [2] for further the details). Regarding AbSS, we have utilized the parameterization proposed in [3]: external archive maximum size of 100 nondominated solutions, the size of the initial set P is 20, the number of iterations in the improvement method is 5 (polynomial mutation with a distribution index equal to 10), SBX crossover (solution combination method) also with a distribution index equal to 10, and the size of $RefSet_1$ and $RefSet_2$ as well is 10. Both cMOGA and AbSS stop when 25,000 function evaluations have been computed. It is important to note that 25,000 evals $\times$ 5 simulations/eval means that DFCN has been optimized over 125,000 different network instances.

4.3 Results

Let us now begin with the analysis of the results, which are presented in Table 1. Since both AbSS and cMOGA are stochastic algorithms and we want to provide the results with statistical confidence, 30 independent runs of each multiobjective optimizer have been performed, as well as t-tests at 95% of significance level (last column of Table 1). The t-test assesses whether the means of two samples are statistically different from each other.

If we consider that the two algorithms are configured for obtaining 100 nondominated solutions at most (maximum archive size), values shown in Table 1 point out that most executions of the optimizers fill up the whole archive. Though AbSS returns a slightly higher number of Pareto optima on average than cMOGA does, the difference is negligible and no statistical confidence exists ("–" symbol in t-test column), thus showing that both optimizers have a similar ability for exploring the search space of DFCNT.

As regards to the Set Coverage metric, we want to clarify that results shown in column "AbSS" correspond to $\mathcal{C}(AbSS, cMOGA)$ whereas those presented in column "cMOGA" are $\mathcal{C}(cMOGA, AbSS)$. As it can be seen in Table 1, AbSS gets larger values for this metric than cMOGA and there exists statistical confidence for this claim (see "+" symbol in the last column). This fact points out that AbSS can find solutions that dominate more solutions of cMOGA than vice versa. However, Set Coverage values are similar in both the cases, what indicates that each algorithm computes high quality solutions that dominate most solutions of the other, but those high quality solutions are in turn nondominated.

Last row in Table 1 presents the results of the Hypervolume metric. They clearly show now that AbSS overcomes cMOGA when considering at the same

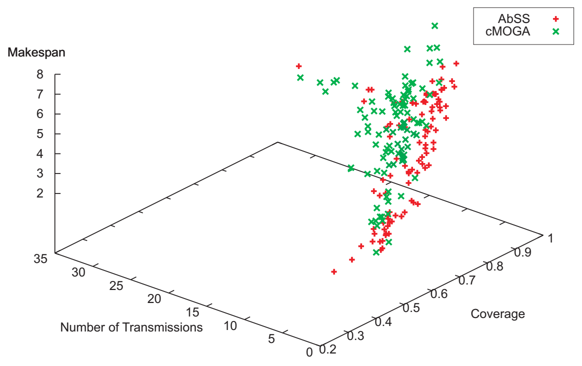


Fig. 3. Two DFCNT fronts from both AbSS and cMOGA

time both convergence and diversity in the resulting Pareto fronts (all this supported with statistical confidence). Since the Set Coverage metric showed that both optimizers were similar in terms of convergence, we can conclude that AbSS is reaching this Hypervolume value because of the diversity in the found Pareto front. That is, the set of nondominated solutions computed by AbSS covers a larger region of the objective space, what is an important feature for actual designs of MANETs. We show an example Pareto front that capture the previous claims in Fig. 3. Regarding coverage, the AbSS front ("+" symbols) is behind (on the right) cMOGA solutions ("×" symbols). With respect to diversity, it also can be seen that there are nondominated solutions from AbSS that reach DFCN configurations where message coverage is around 40% of the stations while cMOGA is not able to get solutions in this region of the objective space. Therefore, using AbSS provides the network designer (decision maker) with a wider set of DFCN parameter settings which ranges from configurations that get a high coverage in a short makespan but using a high bandwidth to those cheap solutions in terms of time and bandwidth being suitable if coverage is not a hard constraint in the network.

5 Conclusions and Future Work

This paper investigated the usage of AbSS, a multiobjective scatter search method, for optimally tuning the DFCN broadcasting strategy for MANETs. The multiobjective problem to be solved is called DFCNT and has three goals: minimizing makespan, maximizing network coverage, and minimizing the network usage. DFCNT has been previously tackled with a cellular multiobjective genetic algorithm called cMOGA.

Three metrics have been used for comparing the optimizers: Number of Pareto optima, Set Coverage, and Hypervolume. Regarding the number of nondominated solutions found, AbSS got a slightly higher number of configurations for DFCN on average than cMOGA, but differences are negligible. Regarding Set Coverage and Hypervolume, resulting values from the metrics claim that solutions from the scatter search approach dominated those obtained with cMOGA (convergence) as well as covered a larger region of the objective space (diversity). From these results, a clear conclusion can be drawn: AbSS is a promising approach for solving DFCNT with advantages over the existing one.

As a future work, we plan to perform more in depth analysis on using AbSS for solving real world MOPs. On the one hand, we also intend to use different scenarios where DFCN has to be tuned and, on the other hand, enlarge the simulation area to a still larger metropolitan network for large cities.

References

1. Hogie, L., Guinand, F., Bouvry, P.: A Heuristic for Efficient Broadcasting in the Metropolitan Ad Hoc Network. In: 8th Int. Conf. on Knowledge-Based Intelligent Information and Engineering Systems. (2004) 727–733
2. Alba, E., Dorronsoro, B., Luna, F., Nebro, A., Bouvry, P.: A Cellular Multi-Objective Genetic Algorithm for Optimal Broadcasting Strategy in Metropolitan MANETs. In: IPDPS-NIDISC'05. (2005) 192
3. Nebro, A.J., Luna, F., Dorronsoro, B., Alba, E., Beham, A.: AbSS: An Archive-based Scatter Search Algorithm for Multiobjective Optimization. European Journal of Operational Research (2005) Submitted
4. Glover, F.: A Template for Scatter Search and Path Relinking. In: Third European Conf. on Artificial Evolution. Volume 1363 of LNCS. Springer Verlag (1997) 3–54
5. Glover, F., Laguna, M., Martí, R.: Fundamentals of Scatter Search and Path Relinking. Control and Cybernetics **29** (2000) 653–684
6. Glover, F., Laguna, M., Martí, R.: Scatter Search. In: Advances in Evolutionary Computing: Theory and Applications. Springer, New York (2003) 519–539
7. Beausoleil, R.P.: MOSS: Multiobjective Scatter Search Applied to Nonlinear Multiple Criteria Optimization. Eu. J. of Operational Research **169** (2005) 426–449
8. da Silva, C.G., Clímaco, J., Figueira, J.: A Scatter Search Method for the Bi-Criteria Multi-Dimensional {0,1}-Knapsack Problem using Surrogate Relaxation. Journal of Mathematical Modelling and Algorithms **3** (2004) 183–208
9. Nebro, A.J., Luna, F., Alba, E.: New Ideas in Applying Scatter Search to Multiobjective Optimization. In: EMO 2005. LNCS 3410 (2005) 443–458
10. Williams, B., Camp, T.: Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In: Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC). (2002) 194–205
11. Knowles, J., Corne, D.: The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC. (1999) 9–105
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197
13. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Swiss Federal Inst. of Technology (2001)
14. Deb, K., Agrawal, B.: Simulated Binary Crossover for Continuous Search Space. Complex Systems **9** (1995) 115–148
15. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Swiss Federal Institute of Technology (ETH) (1999)
16. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Study. In: PPSN V. (1998) 292–301

Evolutionary Design of OAB and AAB Communication Schedules for Interconnection Networks

Miloš Ohlídal, Jiří Jaroš, Josef Schwarz, and Václav Dvořák

Brno University of Technology, Faculty of Information Technology,
Department of Computer Systems, Božetěchova 2, 612 66 Brno, Czech Republic
phone: +420-541141149, fax: +420-541141270
{Ohlidal, Jarosjir, Schwarz, Dvorak}@fit.vutbr.cz

Abstract. Since chip multiprocessors are quickly penetrating new application areas in network and media processing, their interconnection architectures become a subject of sophisticated optimization. One-to-All Broadcast (OAB) and All-to-All Broadcast (AAB) [2] group communications are frequently used in many parallel algorithms and if their overhead cost is excessive, performance degrades rapidly with a processor count. This paper deals with the design of a new application-specific standard genetic algorithm (SGA) and the use of Hybrid parallel Genetic Simulated Annealing (HGSA) to design optimal communication algorithms for an arbitrary topology of the interconnection network. Each of these algorithms is targeted for a different switching technique. The OAB and AAB communication schedules were designed mainly for an asymmetrical AMP [15] network and for the benchmark hypercube network [16] using Store-and-Forward (SF) and Wormhole (WH) switching.

1 Introduction

With parallel and distributed computing coming of age, multiprocessor systems are more frequently found not only in high-end servers and workstations, but also in small-scale parallel systems for high performance control, data acquisition and analysis, image processing, networking processors, wireless communication, and game computers. The design and optimization of hardware and software architectures for these parallel embedded applications have been an active research area in recent years. For many cases it is better to use several small processing nodes rather than a single big and complex CPU. Nowadays, it is feasible to place large CPU clusters on a single chip (multiprocessor SoCs, MSoCs), allowing both large local memories and the high bandwidth of on-chip interconnect.

One of the greatest challenges faced by designers of digital systems is optimizing the communication and interconnection between system components. As more and more processor cores and other large reusable components have been integrated on single silicon die, a need for a systematic approach to the design of communication part has become acute. One reason is that buses, the former main means to connect the components, could not scale to higher numbers of communication partners. Recently the research opened up in Network on Chip (NoC) area, encompassing the interconnection/communication problem at all levels, from physical to the architectural to the OS and application level [1].

Presently, there are many different interconnection network topologies for general purpose multiprocessors, but new networks for specific parallel applications can still be created. Whereas the lower bounds on the time complexity of various group communications (in terms of required number of communication steps) can be mathematically derived for any network topology and the given communication pattern, finding a corresponding schedule of communication is more difficult and in some cases it is not known as yet. The rest of the paper addresses the quest for an optimal communication schedule based on evolutionary algorithms, provided that network topology and a communication pattern are given.

2 Models of Communications

Communications between two partners (p2p) or among all (or a subset) of partners engaged in parallel processing have a dramatic impact on the speedup of parallel applications. Performance modelling of p2p and group communications is therefore important in design of application-specific systems. A p2p communication may be random (input data dependent) as far as source-destination pair or a message length is concerned. However, in many parallel algorithms we often find certain communication patterns, which are regular in time, in space, or in both time and space; by space we understand spatial distribution of processes on processors. Communications taking place among a subset or among all processors are called group or collective communications. Examples of these may serve One-to-All Broadcast (OAB), All-to-All Broadcast (AAB), One-to-All Scatter (OAS, a private message to each partner), All-to-One Gather (AOG), All-to-All Scatter (AAS), permutation, scan, reduction and others [2]. Provided that the amount of computation is known, as is usually true in case of application-specific systems, the only thing that matters in obtaining the highest performance are group communication times.

The simplest time model of communication uses a number of communication steps (rounds): point-to-point communication takes one step between adjacent nodes and a number of steps if the nodes are not directly connected.

Two types of switching are used in this article. The first one is distance-sensitive Store-and-Forward (SF). Each intermediate node on the path firstly receives the whole message and then sends it to adjacent node in the next possible communication step. The second type of switching is called wormhole (WH) switching. Here several p2p messages between source-destination pairs, not necessarily neighbours can proceed concurrently and can be combined into a single step if their paths are disjoint. Of course, for simplicity, we assume no contention for channels and no resulting delays. An example of these switching techniques is shown in Fig. 1.

Further, we have to distinguish between unidirectional (simplex) channels and bidirectional (half-duplex, full-duplex) channels. The number of ports that can be engaged in communication simultaneously (1-port or all-port models of routers) has also an impact on the number of communication steps and communication time, as well as if nodes can combine/extract partial messages with negligible overhead (combining model) or can only retransmit/consume original messages (non-combining model).

We use all-port non-combining model in our experiments. The goal was to find communication algorithms whose time complexity is as close as possible to mathematically derived lower bounds on number of communication steps.

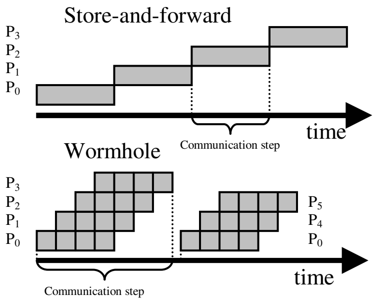


Fig. 1. Basic type of switching techniques

In our experimental runs mostly the well known hypercube [16] and AMP network [15] topologies were tested, see Fig. 2. Optimal schedules for the former topology are known and can therefore be used to evaluate quality of used algorithms; the feature of the latter topology (for which optimal schedules are unknown) is that the number of nodes with degree *d* that can be connected in a network is maximum.

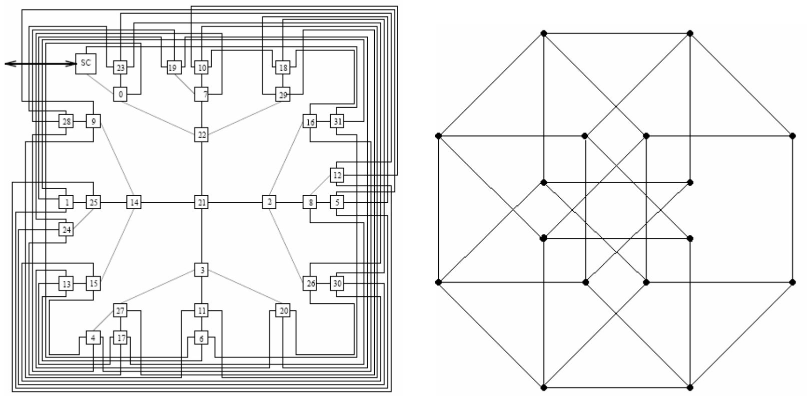


Fig. 2. 32 processors AMP topology and 16 processors hypercube topology

3 Discrete Optimization Algorithms

Combinatorial search and optimization techniques in general are characterized by quest for a solution to a problem from among many potential solutions. For many search and optimization problems, exhaustive search is infeasible and some form of guided search is undertaken instead. In addition, rather than only the best (optimal) solution, a good non-optimal solution is often sought.

3.1 Standard Genetic Algorithm (SGA)

A genetic algorithm [3] is a powerful, domain-independent search technique. SGA is a population-based computational model that uses selection and recombination operators to generate new samples in the search space. A chromosome, consisting of genes, represents one encoded solution from the search space. The values of genes are referred to as alleles. The chromosomes form population, which changes through the evolution process. The reproduction process is performed in such a way that chromosomes, which represent better solutions, are given more chances to reproduce than those chromosomes, which represent poorer solutions. The fitness function (a measure of quality) of chromosomes is defined in the frame of the population. The fitness function is applied to genotype (chromosomes) for evaluating phenotype (decoded form of the chromosome).

One point crossover and integer bound mutation were used as recombination operators and tournament selection as selection operator.

3.2 Hybrid Parallel Genetic Simulated Annealing (HGSA)

HGSA [7] is a hybrid method that uses parallel Simulated Annealing (SA) [10] with the operations used in standard genetic algorithms [8]. In the proposed algorithm, several SA processes run in parallel. After a number of steps (after every ten iterations of Metropolis algorithm), the crossover is used to produce new solutions.

During communication, which is activated each 10th iteration of Metropolis algorithm, all processes sends their solution to a master. The master keeps one solution for himself and sends one randomly chosen solution to each slave. The selection is based on the roulette wheel, where the individual with the best value of the fitness function has the highest probability of selection.

After communication phase, all processes have two individuals. Now the phase of genetic crossover starts. Two additional children solutions are generated from two parent solutions using double-point crossover. The solution with the best value of the fitness function is selected and mutation is performed: always in case of the parent solution, otherwise with a predefined probability. Mutation is performed by randomly selecting genes and by randomly changing their values. A new solution of each process is selected from the actual solution provided by SA process and from the solution, which was obtained after genetic mutation. The selection is controlled by well-known Metropolis criterion.

4 OAB and AAB Communication Patterns

OAB (One-to-All Broadcast) [4, 5] is a collective communication pattern. In this case, one node (initiator) distributes the same message to all other nodes in the interconnection network. If only node subset takes part in communication, we talk about multicast communication pattern (MC). This communication (as well as OAS [11, 12] with distinct messages to receiving nodes) can be performed by sequentially sending the message to particular nodes. This way is very inefficient because only one node sends the message in each communication step. However we can use a better technique using a broadcast tree when every node that received the message in previous communication step becomes an initiator of new multicast communication. Consequently, the number of informed nodes increases by d^k instead by d, where d is the node degree and k is number of communication steps.

The goal of the proposed evolutionary algorithm is to find such a broadcast tree (communication schedule) that it will be possible to inform all nodes in the minimal number of communication steps. A resulting communication schedule has to be conflict-free, i.e. only one message can be transmitted via the same link in the same step and the same direction.

Optimal communication schedules for OAB communication pattern using store and forward and wormhole switching technique on eight nodes ring topology are shown on the left side of Fig. 3. Broadcast trees are shown on the right side.

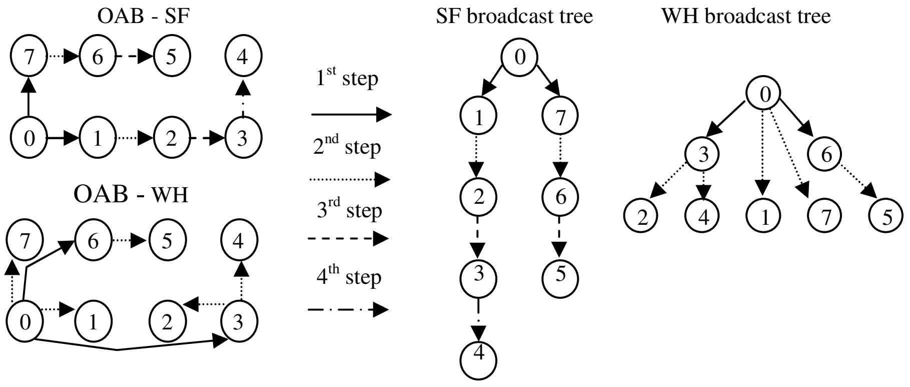


Fig. 3. The optimal OAB schedules for 8 nodes ring topology and the relevant broadcast trees

The lower bounds on the number of communication steps for the all-port hypercube and AMP topology are shown in Table 1. Parameters of the interconnection network in Table 1 are: processors count P, network diameter D, node degree d, bisection width B_C, and average distance d_a.

Table 1. Lower bounds on number of communication steps (all-port models) [13]

	SF hypercube	WH hypercube	SF AMP	WH AMP
OAB	$D\ (= d)$	$\lceil d/\log(d+1)\rceil$	D	$\lceil \log_{d+1} P \rceil$
AAB	$\lceil (P-1)/d\rceil$	$\lceil (P-1)/d\rceil$	$\lceil (P-1)/d\rceil$	$\lceil (P-1)/d\rceil$

5 Design of Algorithms

The goal of proposed algorithms is to find a schedule of a group communication with the number of steps as close as possible to the above lower bounds. The solution of this optimization problem by means of evolutionary algorithms may be decomposed into several phases. In the first phase, it is necessary to choose a suitable encoding of the problem into a chromosome. The second step is a definition of the fitness function, which determines quality of a chromosome. The next phase is design of the input data structure for the evolutionary algorithm. The last phase includes experimental runs of the evolutionary algorithm and search for the best set of its parameters. The choice of parameters should speed-up the convergence of the algorithm and simultaneously minimizes a probability of getting stuck in local minima.

5.1 Solution Encoding

Different encodings were used for each optimization algorithm according to the switching technique. We used an indirect encoding for OAB with wormhole switching optimized by SGA algorithm. Thus a chromosome does not include a decision tree, but only instructions how to create it from chromosome. Any chromosome consists of *P* genes. Every gene corresponds to one destination node. Individual genes include three integer values. The first one is a source node index. The second one determines the shortest path along which the message will by transmitted. The last one is a communication step number when the communication will be performed.

The main disadvantage of this encoding is formation of inadmissible solutions during process of genetic manipulation. We say that a solution is inadmissible if it is not possible to construct correct broadcast tree from it. An example of inadmissible solution can be a case when some node receives a message in a given step from a node that has not received the message yet. That is why admissibility verification has to be carried out for every solution before every fitness function evaluation and if the need be, the restoration will be accomplished. In Fig. 4, a chromosome for wormhole OAB communication patter for the 8-node ring topology is presented.

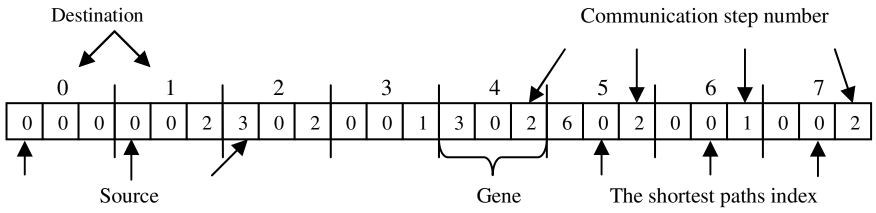


Fig. 4. Encoding of broadcast tree in chromosome for SGA case

Very simply encoding of SF OAB communication pattern has been chosen for HGSA. Every chromosome consists of *P* genes, where *P* is a number of processors in a given topology. The gene's index represents the destination processor for a message. The gene consists of two integer components. The first component is an index of one of the shortest path from source to destination. The second component is a sequence of communication links on the path. Fig. 5 illustrates an example of this encoding. The source processor has index 0. For completeness the chromosome includes also communication from source to source processor, but this communication is not realized. This gene is included only for the easier evaluation of the fitness function.

The main advantage of this encoding is a short chromosome and the absence of inadmissible solutions (every message is transmitted from the source to a destination). The main disadvantage is a large number of possible values of the first gene component. The number of the values rapidly increases with the distance from source to destination as there are more shortest paths between them.

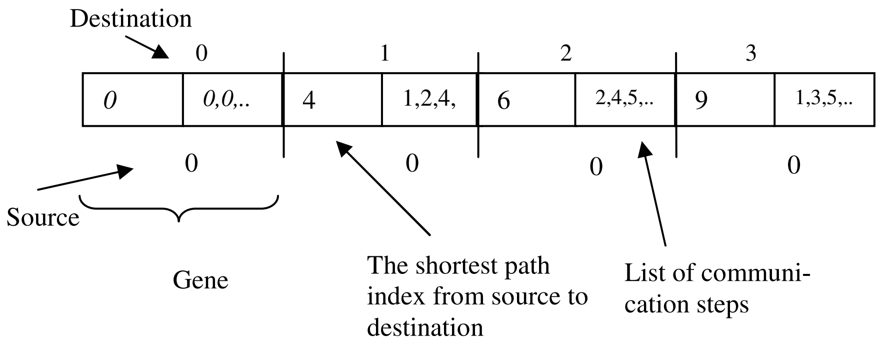

Destination
0
1
2
3
0
0,0,..
4
1,2,4,
6
2,4,5,..
9
1,3,5,..
0
0
0
0
Source
Gene
The shortest path index from source to destination
List of communication steps

Fig. 5. The structure of chromosome of HGSA in case of OAB

The AAB chromosome is an extension of a vector to matrix for both optimization algorithms SGA and HGSA. An AAB chromosome is composed of *P* OAB chromosomes as every processor performs OAB.

5.2 The Fitness Function

The fitness function evaluation is the same for both proposed algorithms. It is based on testing of conflict-freedom. We say that two communication paths are in conflict if and only if they use the same communication link in the same time and in the same direction (see Fig. 6). The fitness function is based on conflict counting. The optimal communication schedule for the given number of communication steps must be conflict-free. If the conflict occurs, the schedule can not be used in real application.

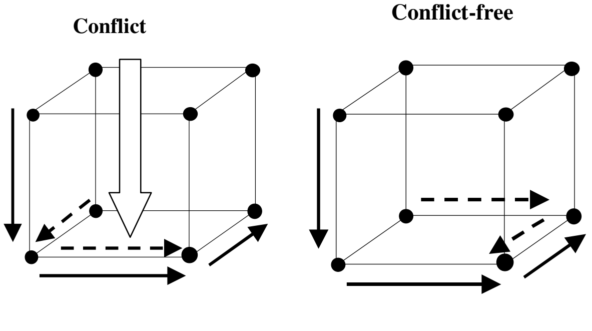


Fig. 6. Conflict in a communication schedule

5.3 The Shortest Paths Algorithm

This algorithm generates all shortest paths and saves them in the operating memory into a specific data structure. The generating algorithm [6] is inspired by the breadth-first search algorithms BFS. BFS is based on the searching a graph, where the source processor is chosen as a root. The edges create a tree used in searching process. A tree is gradually constructed, one level at a time, from the root that is assigned an index of a source node. When a new level of the tree is generated, every node at the lowest level (leaf) is expanded. When a node is expanded, its successors are determined as all its direct neighbours except those, which are already located at higher levels of the tree (it is necessary to avoid cycles). Construction of the tree is finished when a value of at least one leaf is equal to the index of a destination node. Destination leaves' indices confirm the existence of searched paths, which are then stored as sequences of incident node indices.

5.4 Heuristics

In SGA a new heuristic for chromosome restoration was used. The restoration (correction of the broadcast tree) proceeds subsequently in particular communication steps. For every node we check if it receives the message from the node that has already received it in some previous communication step. As far as this condition is not satisfied, the source node of this communication is randomly replaced by a node that already has the message. Further, it is necessary to check already used shortest paths. There is a finite number of the shortest paths from every source to every destination node. If the second gene component (the index path) exceeds this value, the modulo operation will be applied to this gene component.

In HGSA two heuristics are used to speed up the convergence to a sub-optimal solution. They decrease the probability of being trapped in local optima during the execution. The idea is a simple reduction of the path length. The first heuristics is used after the initialization of HGSA and then after each application of Metropolis algorithm. The length of the path from the source to the destination node has some value. If the end node occurs in another gene with a smaller length, than the length and the path in the original gene are changed accordingly.

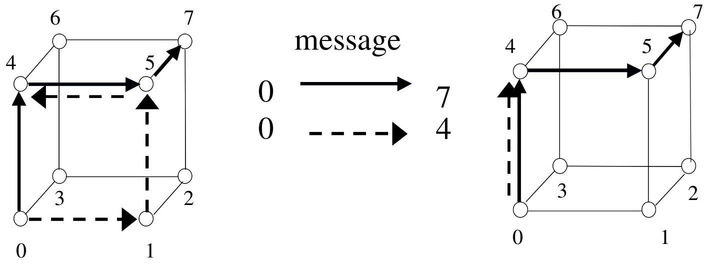


Fig. 7. Reduction of shorter path according to longer path

The second heuristics is used in all surveyed collective communications. It removes using proper setting of the communication step for several nodes incident with the examined path. It means really to endeavour after suspending message in the node during the usage of the same link by another message. However, this changing must not increase the number of the communication steps of the optimal schedule.

In the case that above presented way doesn't lead to improvement, it tries other way and it is endeavor after the fastest sending message from source to destination.

6 Experimental Results

Both sequential SGA and parallel HGSA have been implemented in C/C++. They use only standard C and C++ libraries to ensure good portability. HGSA implementation uses MPI [9] routines for message passing and can therefore be compiled and run on any architecture (clusters of workstations, MPPs, SMPs, etc.) for which an implementation of MPI standard is available.

The proposed algorithms were verified on some multiprocessor topologies (e.g. Midimew, K-Ring...). Two topologies were examined most intensively, namely five cases of hypercubes and five cases of AMP network topologies were used. Other topologies were tested only in 8-node configuration.

The theoretical time complexity in terms of a minimal number of communication steps can be derived for all examined topologies. Theoretical lower bounds of tested topologies are shown in Table 2.

Table 2. Theoretical lower bounds of tested topologies

Lower bounds	Hyper-8	Hyper-16	Hyper-32	Hyper-64	Hyper-128
OAB	2	2	2	3	3
AAB	3	4	5	6	7
	AMP-8	AMP-23	AMP-32	AMP-42	AMP-53
OAB	2	2	3	3	3
AAB	2	6	8	11	13
	K-ring	Midimew	Moore	Octagon	Ladder
OAB	2	2	2	2	4
AAB	2	2	3	3	4

Parameters of SGA were set to the same values for all runs, i.e. probability of crossover 70%, probability of mutation 5%. 10 runs of SGA were performed for each topology, whereas the size of population was set on the value, in which success rate was better than 50%.

Parameters of HGSA were set to the same values for all runs too, i.e. 10 computers in the master slave architecture, the length of communication interval between master and slave was each 10's iterations of Metropolis algorithm 10/10 (OAB/AAB), start temperature 100, number of iterations in each temperature phases was 10, gradient of cooling 0.9/0.99 (OAB/AAB). 15 runs of HGSA were performed for each topology.

We counted only the successful completions, i.e. those reaching the global optimum. The success rate of both algorithms (SGA and HSGA) was measured and compared. If we compare success rate (Table 3) of AMP-23 and AMP-32 topology, we see that the success rate is better for more complex topology. While for AMP-23 not rounded time complexity is 1.94 steps, for AMP-32 it is 2.15 steps. The time complexity of optimal communication schedule can not exceed two communication steps in the first case whereas it can be split into three steps in the second case. By comparing not rounded and rounded time complexities we can make a conclusion, that in the case of AMP-32 topology, much more interconnection links remain unused and the evolutionary algorithm has more space to find the optimal schedule. The same abnormality can be seen in some other topologies (hyper-32 and hyper-64). The success rate 100% was achieved for all other examined topologies.

The presented data of HGSA deserves some comments. Firstly, OAB (SF) is quite a simple operation and therefore the algorithm is likely to find an optimal solution even for larger architectures. Optimal solutions have already been found for topologies with up to 32 processors and acceptable results have been attained for AAB. A further improvement of these results can be expected in the future, because number of experiments, which could be carried out so far, was limited by the overall run time required for optimization (many hours if optimal solutions are sought). On the other hand, if we need an acceptable solution quickly, the proposed algorithms allow to accept a larger number of communication steps and the solution is found in much shorter time.

Table 3. Success rate in achieving the optimum schedule

	Hyper-8	Hyper-16	Hyper-32	Hyper-64	Hyper-128
SGA – OAB	100%	100%	50%	60%	50%
HGSA - OAB	100%	100%	100%	100%	100%
SGA – AAB	70%	20%	-	-	-
HGSA - AAB	100%	80%	-	-	-
	AMP-8	AMP-23	AMP-32	AMP-42	AMP-53
SGA – OAB	100%	50%	100%	60%	50%
HGSA - OAB	100%	100%	100%	100%	100%
SGA – AAB	70%	30%	10%	-	-
HGSA - AAB	100%	80%	10%	-	-

7 Conclusions

Optimization of communication schedules by means of the proposed evolutionary algorithms has been successful. Optimal communication schedules achieve the lower bounds of communication steps derived from graph-theoretical properties of interconnection networks. It is evident that optimum schedules can speed-up execution of many parallel programs that use collective communication as a part of their algorithm.

We have tested two types of evolutionary algorithms. The first one is standard genetic algorithm SGA and the second one HGSA is a composition of parallel simulated annealing and the standard genetic algorithm. Both presented algorithms are able to find an optimal schedule of the given communication pattern for arbitrary network topology, each one with sufficient efficiency.

The future work will be focused on the communication patterns OAS and AAS in case of HGSA and OAB, AAB in case of Estimation of Distribution Algorithms (EDA) [14]. We will implement multi-criteria optimization in EDA algorithms (without the need to enter the number of communication steps) and to design and implement more efficient heuristics for HGSA.

Importance and novelty of above goals should be emphasized. Algorithms, which would be able to find all types of collective communication on any regular or irregular topology, were not published so far in spite of a growing importance especially for multiprocessors on chips.

Acknowledgement

This research has been carried out under the financial support of the research project FRVS 2848/2006/G1 "Memetic Evolutionary Algorithms Applied to Design of Group Communication between Processors", FRVS 2983/2006/G1 "EDA Evolutionary Design of Group Communication Schedules", (Ministry of Education) and of the research grant GA102/05/0467 "Architectures of Embedded Systems Networks" (Grant Agency of Czech Republic).

References

1. Jantsch, A., Tenhunen, H.: Networks on Chip, Kluwer Academic Publ., Boston, 2003, ISBN 1-4020-7392-5
2. Gabrielyan, E.: Hersch, R.D.: Network′s Liquid Throughput: Topology Aware Optimization of Collective Communication. Unpublished work, 2003
3. Goldberg, D.: Genetics Algorithms in Search, Optimization, and Machine Learning, Addision-Wesley Publishing Company, 1989
4. Defago, X., Schiper, A., Urban, P.: Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey, technical report DSC/2000/036, 2003
5. Duato, J., Yalamanchili, S.: Interconnection Networks – An Engineering Approach, Morgan Kaufman Publishers, Elsevier Science, 2003
6. Staroba J.: Parallel Performance Modelling, Prediction and Tuning, PhD. Thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Rep., 2004
7. Ohlídal, M., Schwarz, J.: Hybrid parallel simulated annealing using genetic operations, In: Mendel 2004 10th Internacional Conference on Soft Computing, Brno, CZ, FSI VUT, 2004, pp. 89-94
8. Goldberg D. E.: A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing, Complex Systems, 1990, pp. 445–460.
9. http://www-unix.mcs.anl.gov/mpi
10. Kita, H.: Simulated annealing. Proceeding of Japan Society for Fuzzy Theory and Systems, Vol. 9, No. 6, 1997
11. Jaroš, J., Dvořák, V.: Speeding-up OAS and AAS Communication in Networking System on Chips, In: Proc. of 8th IEEE Workshop on Design and Diagnostic of Electronic Circuits and Systems, Sopron, HU, UWH, 2005, pp. 206-210
12. Jaroš, J., Ohlídal, M., Dvořák, V.: Evolutionary Design of Group Communication Schedules for Interconnection Networks, In: Proceedings of 20th International Symposium of Computer and Information Science, Berlin, DE, Springer, 2005, s. 472-481

13. Dvořák, V.: Scheduling Collective Communications on Wormhole Fat Cubes, In: Proc. of the 17th International Symposium on Computer Architecture and High Performance Computing, Los Alamitos, US, IEEE CS, 2005, pp. 27-34
14. Pelikan, M., Goldberg, E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occams Razor.
15. Chalmers, A.-Tidmus, J.: Practical Parallel Processing. International Thomson Computer Press, 1996
16. http://www.sgi.com/products/remarketed/origin/pdf/hypercube.pdf

A Multiagent Algorithm for Graph Partitioning

Francesc Comellas and Emili Sapena[*]

Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya,
Avda. Canal Olímpic s/n, 08860 Castelldefels, Catalonia, Spain
comellas@ma4.upc.edu
http://www-ma4.upc.edu/~comellas/

Abstract. The k-cut problem is an NP-complete problem which consists of finding a partition of a graph into k balanced parts such that the number of cut edges is minimized. Different algorithms have been proposed for this problem based on heuristic, geometrical and evolutionary methods. In this paper we present a new simple multiagent algorithm, *ants*, and we test its performance with standard graph benchmarks. The results show that this method can outperform several current methods while it is very simple to implement.

1 Introduction

Graph partitioning is a problem which appears in many different applications such as VLSI design, data-mining, finite elements and communication in parallel computing. In the latter case, for example, the subdomains are mapped to processors and to avoid bottlenecks, the assignment should be as uniform as possible and the data exchange between processors minimized. In terms of graphs, the goal is to find a balanced k-partition of a graph $G = G(V, E)$ with a minimal number of cut edges separating the sets from each other (k-cut). As this is an NP-complete problem [5], for graphs with a large order we cannot be sure we can find an optimal solution in a reasonable computation time: when the size of the graph increases, the execution time of an algorithm capable of solving the problem can be assumed to grow exponentially. Therefore the problem is practically unsolvable for most networks and for this reason heuristic and probabilistic methods are implemented to obtain solutions close to the optimal in a reasonable time. The only way to guarantee the optimal solution is an exhaustive search. Nevertheless, this is only applicable to very simple problems in which the number of nodes is small.

Given the importance of the k-cut problem, there is a large literature proposing different algorithms. For example, in [6] an heuristic method is introduced which can find, for the general problem, approximate solutions in time $O(|V|^{k^2})$ and in [9] the algorithm proposed can find solutions within a factor of $(2 - 2/k)$

[*] Research supported by the Ministerio de Educación y Ciencia, Spain, and the European Regional Development Fund under project TIC2002-00155.

of the optimal k-cut requiring $|V|-1$ max-flow computations. Recent more efficient methods are based on multilevel paradigms [1, 12], in some cases combined with evolutionary algorithms [10, 8].

In this paper we use a multiagent algorithm, *ants*, for the k-cut problem. This algorithm has proved useful in coloring and frequency assignment problems [3, 2]. The method is simple to understand and to implement and the results obtained with standard benchmarks show that it can outperform current algorithms.

2 Graph Partitioning

Let $G = G(V, E)$ be an undirected graph where V is the vertex set and E the edge set. Although in the general partitioning problem both vertices and edges can be weighted, here as in most of the literature, they are given unit weights. A partition of the graph is a mapping of V into k disjoint subdomains S_i such that the union of all subdomains is V, i.e. $\bigcup_{i=1}^{k} S_i = V$. The cardinality of a subdomain is the number of vertices in the subdomain S_i, and the set of inter-subdomain or cut edges (i.e. edges cut by the partition) is denoted by E_c and referred to as the k-cut. The objective of graph partitioning is to find a partition which evenly balances the cardinalities of each subdomain whilst minimizing the total number of cut edges or cut-weight, $|E_c|$. To evenly balance the partition, the cardinality of the optimal subdomain is given by $|S_{opt}| = \left\lceil \frac{|V|}{k} \right\rceil$. The graph partitioning problem can then be specified as: find a partition of G such that $|E_c|$ is minimised subject to the constraint that $|S_{opt}| - 1 \leq |S_i| \leq |S_{opt}|$ for $1 \leq i \leq k$. In this paper we find partitions with perfect balance.

3 The *ants* Algorithm

The partitioning or k-cut problem described in the previous section, as many other problems in graph theory, is an NP-complete problem [5]. Efficient algorithms for this problem are known only for very particular classes of graphs. When exact methods are not possible, sometimes it is sufficient to obtain an approximate solution with a fast and easy to implement method. This is the case of simulated annealing, genetic algorithms, neural networks, ant colony based systems, or multiagent methods like the one described here.

To implement any of these optimization methods we need a way to encode the problem which has to be solved, and a system to quantify how "good" a solution is. In our case a possible solution may be described using a list such that each position is associated to a vertex of the graph and its value to a color which represents the subdomain to which it belongs. The global cost function simply counts the number of times that an edge joins vertices of different colors. We use also a local cost function associated to each vertex defined as the ratio between the number of neighbors that have different colors to the total number of neighbors.

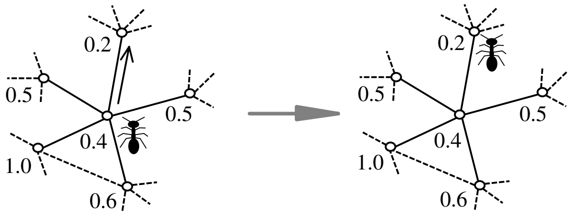


Fig. 1. Movement of an ant towards the worst local node

The *ants* algorithm is a multiagent system based on the idea of parallel search. Unlike other algorithms with a similar name which are generically known as ant-colony optimization, our algorithm does not use "pheromones" or local memory. Thus it is faster and easier to implement. A generic version of the algorithm was proposed in [3]. The mechanism of the algorithm is as follows: Initially the graph is k vertex colored at random keeping the number of vertices for each color balanced. A given number of agents, which we call ants, is placed on the vertices also at random. Then the ants move around the graph and change the coloring according to a local optimization criterion: At a given iteration each ant moves from the current position to the adjacent vertex with the lowest local cost, i.e. the vertex with the greatest number of constraints (neighbors of a different color) and replaces its color with a new color which increases the local cost. At the same time, and to keep the balance, the algorithm chooses, from a set of s random vertices, one with the lowest value of the local cost function -from those which have the new color- and changes its color to the old color. After these color changes, the local cost function is updated for the two chosen vertices and their neighhors. The value of s is not critical, and for our tests we considered $s = 100$. The actions are randomly repeated for each ant. An essential characteristic of the algorithm comes precisely from the stochastic nature of the changes performed. The agent or ant moves to the worst adjacent vertex with a probability p_m (it moves randomly to any other adjacent vertex with probability $1 - p_m$), and assigns the best color, under probability p_c (otherwise it assigns any color at random). Both probabilities are adjustable parameters and allow the algorithm to escape from local minima and obtain partitions with k-cuts close to the optimal. The process is repeated until a solution fulfilling all the constraints is found or the algorithm converges. The number of ants in the algorithm is another adjustable parameter that should increase with the diameter of the graph (the maximum of the distances between pairs of vertices).

In the same way as in an insect colony the action of different agents with simple behaviors gives rise to a structure capable of carrying out complicated tasks, the algorithm presented here, which is based on a series of simple local actions that might even be carried out in parallel, can obtain restrictive graph partitions. Note that our algorithm is not a simple sum of local searches, as they would quickly lead to a local solution. The probabilities p_m and p_c play an important role in avoiding these minima, however their values are not critical and affect mainly the convergence time of the algorithm which is shorter for larger values of p_m and p_c as the index of local improvement at each iteration increases.

An outline of the *ants* algorithm is shown here in pseudocode.

ANTS ALGORITHM:

```
Initialize
    $n$ (number of ants), $k$, $p_m$, $p_c$
    Color each vertex of the graph at random forming $k$ balanced sets
    Put each ant on a randomly chosen vertex
    For all vertices
        Initialize local_cost_function
    End for
    Initialize global_cost_function
    best_cost := global_cost_function
While (best_cost > 0) do
    For all ants
        If (random < $p_m$)
            Move the ant to the worst adjacent vertex
        Else
            Move randomly to any adjacent vertex
        End if
        If (random < $p_c$)
            Change vertex color to the best possible color
        Else
            Change to a randomly chosen color
        End if
        Keep balance (Change a randomly chosen vertex with low local cost
                      from the new to the old color )
        For the chosen vertices and all adjacent vertices
            Update local_cost_function
            Update global_cost_function
        End for
        If (global_cost_function < best_cost )
            best_cost = global_cost_function
        End if
    End for
End while
```

4 Results

We tested the algorithm using a set of benchmark graphs which are available from the Graph Partitioning Archive, a website maintained by Chris Walshaw [13]. The graphs can also be downloaded from Michael Trick's website [11]. Many

Table 1. Best partitions found with *ants* corresponding to perfect balance for 16 subdomains using benchmark graphs[13]. We provide for the graphs of reference the total of vertices and edges, optimal subdomain size, cut size, new cut size and the algorithm used to find the old cut size. Boldface denotes those values for which the *ants* algorithm has outperformed the known result. Algorithms: Ch2.0, CHACO, multilevel Kernighan-Lin (recursive bisection), version 2.0 (October 1995) [7]. J2.2, JOSTLE, multilevel Kernighan-Lin (k-way), version 2.2 (March 2000) [14]. iJ, iterated JOSTLE, iterated multilevel Kernighan-Lin (k-way) [12]. JE, JOSTLE Evolutionary, combined evolutionary/multilevel scheme [10].

Graph	vertices	edges	domain size	cut size [13]	new cut size	algorithm
C2000.5	2000	999836	125	923294	**922706**	Ch2.0
C4000.5	4000	4000268	250	3709887	**3708532**	Ch2.0
DSJC125.1	125	736	8	524	**522**	iJ
DSJC1000.1	1000	40629	63	43078	**43001**	Ch2.0
DSJC1000.5	1000	249826	63	229362	**228850**	Ch2.0
jean	80	254	5	161	161	Ch2.0
flat1000_50_0	1000	245000	63	224403	**224378**	Ch2.0
flat1000_60_0	1000	245830	63	225546	**225183**	Ch2.0
flat1000_76_0	1000	246708	63	226371	**225962**	Ch2.0
le450_5a	450	5714	29	4063	**4030**	JE
le450_5b	450	5734	29	4065	**4055**	iJ
le450_5c	450	9803	29	7667	**7656**	iJ
le450_15a	450	8168	29	5636	**5619**	iJ
le450_15b	450	8169	29	5675	**5641**	iJ
le450_15c	450	16680	29	13512	**13509**	iJ
le450_15d	450	16750	29	13556	**13550**	iJ
le450_25a	450	8260	29	5325	**5302**	J2.2
le450_25b	450	8263	29	5041	**5037**	JE
le450_25c	450	13343	29	13457	**13456**	iJ
le450_25d	450	17425	29	13584	**13539**	iJ
miles500	128	1170	8	771	**770**	JE
miles750	128	2113	8	1676	**1673**	iJ
miles1000	128	3216	8	2770	**2768**	iJ
miles1500	128	5198	8	4750	4750	J2.2
mulsol.i.1	197	3925	13	3275	**3270**	Ch2.0
mulsol.i.5	185	3973	12	3371	**3368**	Ch2.0
myciel4	23	71	2	64	64	J2.2
myciel5	47	236	3	205	205	J2.2
myciel7	191	2360	12	1921	**1920**	iJ
queen5_5	25	160	2	151	151	J2.2
queen8_8	64	728	4	632	632	Ch2.0
queen8_12	96	1368	6	1128	1128	Ch2.0
queen12_12	144	2596	9	2040	**2020**	iJ
queen16_16	256	6320	16	4400	4400	J2.2

of these graphs were collected together for the second DIMACS implementation challenge “NP Hard Problems: Maximum Clique, Graph Coloring, and Satisfiability”, see [4]. For the test we considered the most difficult case of perfect balance and choose a partition into 16 sets.

The number of ants ranged from 3 to 9 depending on the order of the graph. The probabilities p_m and p_c were 0.9 and 0.85, respectively. We repeated the algorithm 20 times for each graph (80 times for small graphs) and recorded the best solutions. Each algorithm run takes around one minute, for a C program (400 lines) compiled on a PC Pentium IV 2.8 GHz under Windows XP using Dev-C++.

Most of the improvements were on results obtained previously with CHACO (Ch2.0), a multilevel Kernighan-Lin (recursive bisection) [7] and iterated JOSTLE (iJ), an iterated multilevel Kernighan-Lin (k-way)[12]. In three cases we improved on results obtained with JOSTLE Evolutionary (JE), a combined evolutionary/multilevel scheme [10], and in six more cases we matched or outperformed results obtained with JOSTLE (J2.2), a multilevel Kernighan-Lin (k-way)[14]. In our experiments, the algorithm *ants* obtains better solutions for the coloring test suite (16 subdomains, perfect balance) of graphs considered in [12] in 27 cases and an equivalent solution in 7 cases (out of the 89 graph instances).

5 Conclusion

The results show that our implementation of the multiagent algorithm *ants* for the graph partitioning problem provides, for the balanced case, a new method which complements, and even outperforms, known techniques. Given the simplicity of the algorithm and its performance in the difficult case of balanced sets, it is a promising method for graph partioning in the non-balanced cases. Note also that adapting the algorithm for graphs with weighted vertices and edges would be straightforward.

References

1. R. Baños, C. Gil, J. Ortega, and F. G. Montoya. Multilevel heuristic algorithm for graph partitioning. In 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization. *Lecture Notes in Comput. Sci.* **2611** pp. 143–153 (2003).
2. J. Abril, F. Comellas, A. Cortés, J. Ozón, M. Vaquer. A multi-agent system for frequency assignment in cellular radio networks. *IEEE Trans. Vehic. Tech.* **49**(5) pp. 1558–1565 (2000).
3. F. Comellas and J. Ozón,. Graph coloring algorithms for assignment problems in radio networks. Proc. Applications of Neural Networks to Telecommunications 2, pp. 49-56. J. Alspector, R. Goodman and T.X. Brown (Eds.), Lawrence Erlbaum Ass. Inc. Publis., Hillsdale, NJ (1995)
4. The Second DIMACS Implementation Challenge: 1992-1993: NP Hard Problems: Maximum Clique, Graph Coloring, and Satisfiability. Organized by M. Trick (accessed January 8, 2006). `http://dimacs.rutgers.edu/Challenges/index.html`

5. M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness, New York: W.H. Freeman, 1979, ISBN 0-7167-10447.
6. O. Goldschmidt and D.S. Hochbaum. Polynomial algorithm for the k-cut problem. Proc. 29th Ann. IEEE Symp. on Foundations of Comput. Sci., IEEE Computer Society, 444-451. (1988).
7. B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In S. Karin, editor, Proc. Supercomputing '95, San Diego. ACM Press, New York, 1995.
8. P. Korošec, J. Šilc, B. Robič. Solving the mesh-partitioning problem with an ant-colony algorithm. *Parallel Computing* **30**(5-6) pp. 785–801 (2004).
9. H. Saran and V. Vazirani. Finding k-cuts within twice the optimal. Proc. 32nd Ann. IEEE Symp. on Foundations of Comput. Sci., IEEE Computer Society, 743–751 (1991)
10. A. J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph partitioning. *J. Global Optimization* **29**(2) pp. 225–241 (2004).
11. M. Trick. Graph coloring instances (web page), accessed January 8, 2006. `http://mat.gsia.cmu.edu/COLOR/instances.html`
12. C. Walshaw. Multilevel refinement for combinatorial optimisation problems. *Annals Oper. Res.* **131** pp. 325–372 (2004).
13. C. Walshaw. The graph partioning archive (web page), accessed January 8, 2006. `http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/`
14. C. Walshaw and M. Cross. Mesh Partitioning: a Multilevel Balancing and Refinement Algorithm. *SIAM J. Sci. Comput.* **22**(1) pp. 63–80 (2000).

Tracing Denial of Service Origin: Ant Colony Approach

Chia-Mei Chen, Bing Chiang Jeng, Chia Ru Yang, and Gu Hsin Lai

Department of Information Management,
National Sun Yat-Sen University,
Kaohsiung, 804, Taiwan

Abstract. Denial-of-Service (DoS) attacks with fake source IP addresses have become a major threat to the Internet. Intrusion detection systems are often used to detect DoS attacks. However, DoS attack packets attempt to exhaust resources, degrading network performance or, even worse, causing network breakdown. The proposed proactive approach is allocating the original attack host(s) issuing the attacks and stopping the malicious traffic, instead of wasting resources on the attack traffic.

Ant colony based traceback approach is presented in this study to identify the DoS attack original source IP address. Instead of creating a new function or processing a high volume of fine-grained data, the proposed IP address traceback approach uses flow level information to identify the origin of a DoS attack.

The proposed method is evaluated through simulation on various network environments. The simulation results show that the proposed method can successfully and efficiently find the DoS attack path in various simulated network environments.

Keywords: IP traceback, NetFlow, DoS, Ant algorithm.

1 Introduction

According to a study conducted by the Computer Security Institute in 2003 [1], 90 percent of the 530 surveyed companies had detected computer security breaches in 2003. The same study found that 74 percent acknowledged financial losses due to these security breaches. Although only 47 percent were able to quantify their losses, the financial losses reported by 251 respondents totaled more than $202 million US dollars. However, it is just a proverbial tip of the iceberg. Furthermore, according to the statistics of Dollar Amount of Losses by Type [1], the denial of service (DoS) attack is the second most expensive computer crime among survey respondents with the cost of more than 65 million US dollars.

Nowadays, many organizations use firewall and Intrusion Detection Systems (IDS) to secure their network. If the attacker conducts a DoS attack with a large amount of traffic, the network would still be tied up. Most work in this area has focused on tolerating attacks by mitigation their effects on the victim. Such a passive approach can provide an effective stopgap measure, but does not eliminate the problem nor does it discourage the attackers.

The proactive approach is to find the source of the DoS attack and to cooperate with the internet service provider (ISP) or the network administrators stopping the traffic from the origin. Hence, it can restore normal network functionality, preventing reoccurrences and, ultimately, holding the attackers accountable. However, many network-based DoS attacks use the flaw of TCP/IP to manipulate and falsify the source address in the IP header. Conventional trace methods might not be able to identify the origin as the source address could be spoofed.

The goal of this work is to propose an IP traceback approach to finding out the origin of the DoS attack using the existing traffic flow information, without extra support from the routers. Furthermore, some previous work needs to process a large amount of packets, which may be too costly for detecting DoS attacks. An ant colony based traceback algorithm is proposed, using the traffic flow information as the trace for ants to discover the attack path.

2 Related Work

Savage et al. [2] described and implemented probabilistic packet marking (PPM). When a packet passes through a router, the router determines if marking this packet according to a predefined probability to the IP fragment identification field is facilitated to store the IP Traceback information.

Song and Perrig [3] proposed modifications on Savage's method to further reduce storage requirements by storing a hash of each IP address, instead of the address itself. It assumes that the victim possesses a complete network map of all upstream routers. After edge-fragment reassembly, the method compares the resulting IP address hashes with the router IP address hashes derived from the network map to facilitate attack path reconstruction.

One disadvantage of packet marking approach is that all routers on the attack path are required to support packet marking. In addition, the IP header encoding may have practical restrictions. It negatively impacts users that use fragmented IP datagrams and such encoding might have compatibility issues with the current TCP/IP framework.

Snoeren et al. [4] proposed a hash-based IP traceback scheme, Source Path Isolation Engine (SPIE). As packets traverse the network, digests of the packets get stored in the router. The hash-based IP traceback is predicated on the deployment of SPIE-enhanced routers in place of existing routers in the network infrastructure. But this deployment path was impractical, a SPIE system must be incrementally deployable in the existing network infrastructure without retrofit or 'forklift' upgrade. So the following research focuses on how to implement SPIE on current network infrastructure. Strayer et al. [5] propose the concept of a SPIE Tap Box which is a small, special purpose device that implements the full functionality of the SPIE but without the benefit of access to the router's forwarding engine and internal data structure. Rather, the Tap Box must rely only on the information it can glean by passively tapping the lines into and out the router.

Processing overhead is a drawback of hash-based IP traceback, incurred in every packet when the router to store its digest in the bloom filter. Another drawback is that all the routers must support SPIE.

2.1 Ant Algorithm

Ant algorithms [7] are inspired by the behavior of natural ants and applied to many different discrete optimization problems, such as vehicle routing and resource scheduling. In an Ant algorithm, multiple agents, represented by ants, cooperate with each other using indirect communication mediated by pheromone. The Ant colony algorithm was first introduced to solve the Traveling Salesman Problem (TSP) [6].

A moving ant lays some pheromone (in varying quantities) on the ground, thus marking the path it follows by a trail of this substance. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with a high probability to follow it, then reinforcing the trail with its own pheromone.

A major characteristic of ant algorithms are the viability of autocatalytic processes. A "single-ant" autocatalytic process usually converges very quickly to a bad suboptimal solution. Luckily, the interaction of many autocatalytic processes can lead to rapid convergence to a subspace of the solution space that contains many good solutions, causing the search activity to find quickly a very good solution, without getting stuck in it. In other words, all the ants converge not to a single solution, but to a subspace of solutions; thereafter they go on searching for improvements of the best found solution. Therefore, we believe this feature will be helpful for finding a DoS path.

3 Ant-Colony Approach to DoS Traceback

While an isolated ant moves essentially at random, an ant encountering a previously laid pheromone trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges is a form of autocatalytic behavior where the more the ants are following a trail, the more attractive that trail becomes for being followed. In the proposed IP traceback scheme, we use the average amount of the octets belonging to a DoS attack as the pheromone. Therefore, a router with heavy traffic and more DoS attack flows; more ants will choose it as the next node to move. This will form a positive feedback loop, and finally most ants will follow the same path.

In the initialization phase, ants are positioned on the victim and initial values for pheromone trail intensity are set on each router. When an ant starts from the victim, it will use the topology information to find out all the neighbor routers, and then read the flow information and the pheromone trail of neighbor nodes to compute the probability. Then choose the next router to move to with the probability; this procedure is repeated recursively until it reaches the boundary routers of the monitored network.

When all the ants complete their travels, we use the information gathered by ants to recompute the pheromone trail intensity. Then the next cycle starts with new pheromone trail intensity, until we find most of ants converge to the same path. In the following section, we will describe the details of the proposed IP traceback scheme.

When the IDS on the victim's network detects a DoS attack with spoofed source(s), it could further analyze the packets of the DoS attack and find out the suspected spoofed source IP address list. The proposed solution could take the victim host as the starting point and perform the IP traceback. The detail of the ant colony based DoS path traceback is described as follows.

At the initial stage, each network node uses the amount of total octets sent in duration as f_i and an initial value $\tau_i(t)$. The flow information is selected to determine the probability when an ant chooses a path.

$$p_i(t) = \frac{[\tau_i(t)]^{\alpha} \cdot [f_i]^{\beta}}{\sum\limits_{i \in neighbor} [\tau_i(t)]^{\alpha} \cdot [f_i]^{\beta}}$$

where f_i is the total octets sent in duration of router j, and τ_i (t) is the intensity of pheromone trail on router i at time t.

Figure 1(a) illustrates the case that the ants arrive at Router4, the probability of their next move is determined based on the flow information of the neighbor routers. We assume that the total octet sent from Router5 is 2000, Router6 is 5000 and Router7 is 3000. Therefore, the probability of choosing Router5 is 20%, Router6 is 50% and Router7 is 30%. Figure 1(b) shows the probabililty of the next move to each neighbor router. More ants would choose the path with more flow, as a DoS attack generates lots of flows.

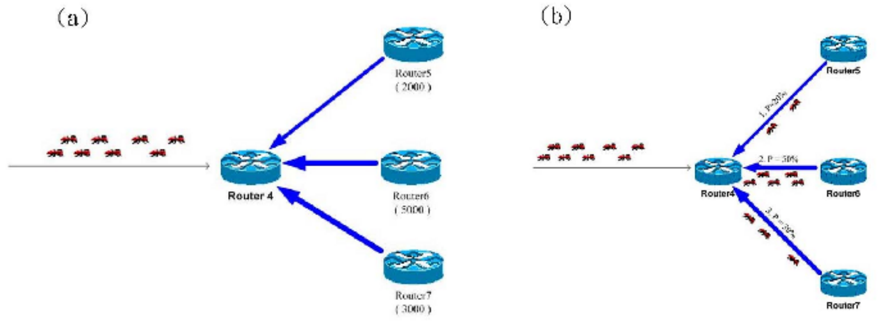


Fig. 1. (a) the flow of Router 4; (b) the probability of selecting the next step

While exploring the network, each ant keeps track of the path and the number of DoS flows. The above procedure is repeated tracing back to the upstream routers until the ant reaches a boundary router of the monitored network. The intensity of pheromone trail is revised after all the ants complete their route from the victim to a boundary router. The path information obtained by each ant is used to calculate $\Delta \tau_{ij}(t, t+1)$:

$$\Delta\tau_i^k = \frac{Q_k}{L_k},$$

where Q_k is the total amount of the octets belonging to the DoS attack on the k-th ant's path and L_k is the length of the k-th ant's path. $\Delta\tau_i(t,t+1)$ is the summation of the pheromone laid by all the ants, expressed below.

$$\Delta\tau_i(t,t+1) = \sum_{k-1}^{m} \Delta\tau_i^k(t,t+1),$$

where $\Delta\tau_i^k(t,t+1)$ is the quantity per unit of length of pheromone laid on router i by the k-th ant between time t and t+1, so the more ants pass through the edge, the more pheromone will be laid on edge. The change of pheromone results in positive feedback -- the more ants are following a path, the more attractive that path becomes for being follow.

The intensity of pheromone on router *i* can be revised once $\Delta\tau_i(t,t+1)$ is obtained and is formulated as below.

$$\tau_i(t+1) = \rho\cdot\tau_i(t) + \Delta\tau_i(t,t+1),$$

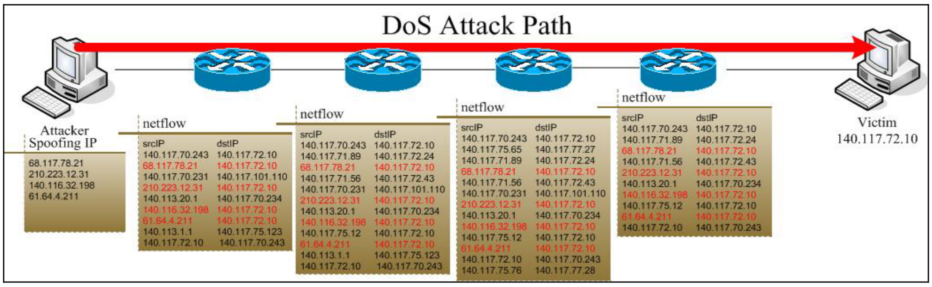
where ρ is a coefficient such that (1-ρ) represents the evaporation of pheromone.

Each time all ants complete one iteration (cycle), the intensity of pheromone on each router will be recalculated based on the above equation. Following the above illustration shown in Figure 1, there would be more pheromone accumulated on Router 7 which results in attracting more ants on Router 4 to choose Router 7 in the following iterations, as Router 7 is on the DoS attack path.

The ant traceback process iterates until the tour counter reaches the user-defined number of cycles or all ants make the same tour. The DoS attack path is constructed by following the biggest probability of the upstream upstream router. In other words, the proposed traceback follows the path chosen by most ants to find the DoS attack path.

3.1 NetFlow

NetFlow is a traffic profile monitoring technology [8] and could provide vital information for DoS traceback. If the packet belongs to an existent flow, traffic statistics of the corresponding flow will be increased, otherwise a new flow entry will be created.



Attacker Spoofing IP:

68.117.78.21
210.223.12.31
140.116.32.198
61.64.4.211

netflow:

srcIP	dstIP
140.117.70.243	140.117.72.10
68.117.78.21	140.117.72.10
140.117.70.231	140.117.101.110
210.223.12.31	140.117.72.10
140.113.20.1	140.117.70.234
140.116.32.198	140.117.72.10
61.64.4.211	140.117.72.10
140.113.1.1	140.117.75.123
140.117.72.10	140.117.70.243

netflow:

srcIP	dstIP
140.117.70.243	140.117.72.10
140.117.71.89	140.117.72.24
68.117.78.21	140.117.72.10
140.117.71.56	140.117.72.43
140.117.70.231	140.117.101.110
210.223.12.31	140.117.72.10
140.113.20.1	140.117.70.234
140.116.32.198	140.117.72.10
140.117.75.12	140.117.72.10
61.64.4.211	140.117.72.10
140.113.1.1	140.117.75.123
140.117.72.10	140.117.70.243

netflow:

srcIP	dstIP
140.117.70.243	140.117.72.10
140.117.75.65	140.117.77.27
140.117.71.89	140.117.72.24
68.117.78.21	140.117.72.10
140.117.71.56	140.117.72.43
140.117.70.231	140.117.101.110
210.223.12.31	140.117.72.10
140.113.20.1	140.117.70.234
140.116.32.198	140.117.72.10
140.117.75.12	140.117.72.10
61.64.4.211	140.117.72.10
140.117.72.10	140.117.70.243
140.117.75.76	140.117.77.28

netflow:

srcIP	dstIP
140.117.70.243	140.117.72.10
140.117.71.89	140.117.72.24
68.117.78.21	140.117.72.10
140.117.71.56	140.117.72.43
210.223.12.31	140.117.72.10
140.113.20.1	140.117.70.234
140.116.32.198	140.117.72.10
140.117.75.12	140.117.72.10
61.64.4.211	140.117.72.10
140.117.72.10	140.117.70.243

Fig. 2. DoS NetFlow records

A conceptual diagram of DoS NetFlow records is shown in Figure 2. The NetFlow records exported by the routers along the DoS attack path will contain the DoS flows whose source IP addresses are the spoofed ones. Such a feature is used to determine if a router is on the DoS attack path and a traceback task can be initiated to find the source of the DoS attack.

4 Performance Evaluation

We verify the proposed solution by implementing the proposed system and evaluating the performance by simulation. A simulated network with NetFlow-enable routers is deployed, as the proposed DoS traceback solution uses the flow-level information to perform the traceback.

4.1 System Architecture

The proposed system architecture contains two major components in a monitored network: the flow management component and traceback module, as shown in Figure 3. The flow management component collects the flow information of the routers in the monitored network in support of the traceback module querying the related flow information. The traceback module performs the traceback based on the flow information.

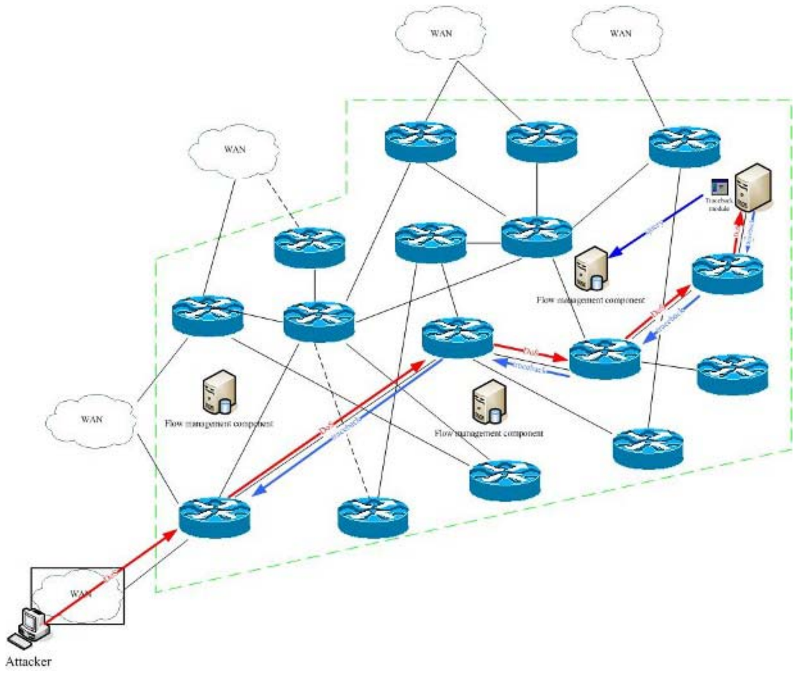

WAN
WAN
WAN
WAN
WAN
Traceback module
Flow management component
Flow management component
Flow management component
DoS
traceback
Attacker

Fig. 3. The proposed system architecture

The flow management component collects the flow based attributes. The open-source tools, Scientific Linux [9], flow-tools [10], STREAM [11], are adopted in this research to achieve the above NetFlow management purpose.

The proposed IP traceback scheme is based on ant algorithm and use NetFlow logs to simulate the IP traceback process. Using artificial ants to explore the network and collect information about the denial-of-service attacks to forecast the possible attack path and traceback to the origin of the DoS attack.

4.2 Experimental Results

A simulated network environment is illustrated in Figure 4, deployed by VMware Workstation [12]. Zebra [13], a routing freeware managing TCP/IP based routing protocol, is adopted to simulate the routers in the experimental environment running on FreeBSD.

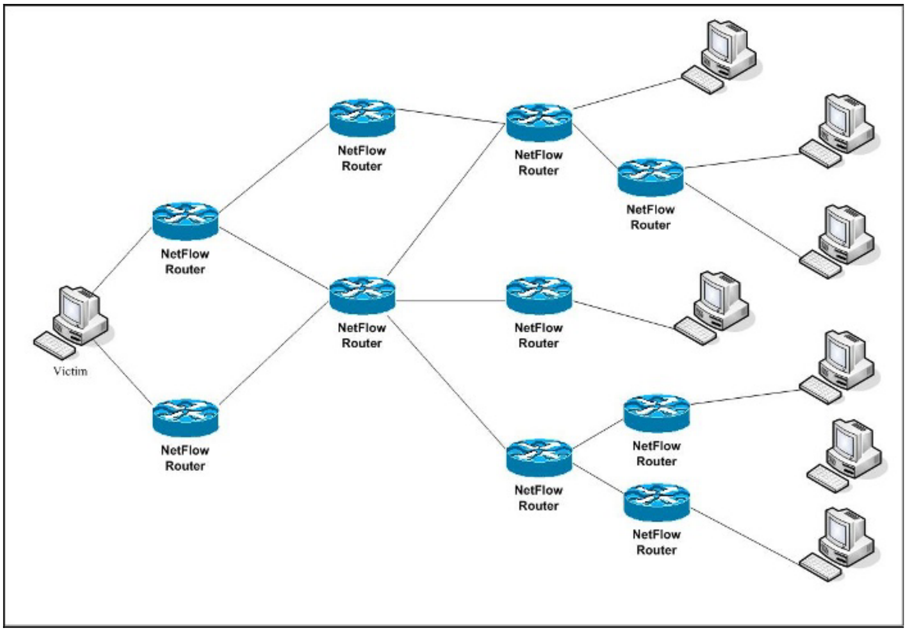


Fig. 4. The simulated network environment

In order to simulate the NetFlow function on Cisco equipments, we use fprobe [14][1] to monitor the traffic and periodically export NetFlow record to proposed NetFlow management. In the simulated network environment, we use Harpoon [14] to generate realistic network traffic which can generate TCP and UDP packet flows and simulates the temporal and spatial characteristics as measured at the routers in a live

[1] A libpcap-based tool collects network traffic data and emits it as NetFlow flows to the specified collector.

environment. Hping [15] is selected to simulate SYN Flood attack with IP spoofing. Hping, a complex ping-based program, can send the customized pings to the remote hosts and networks. The simulated attack scenario is illustrated in Figure 5.

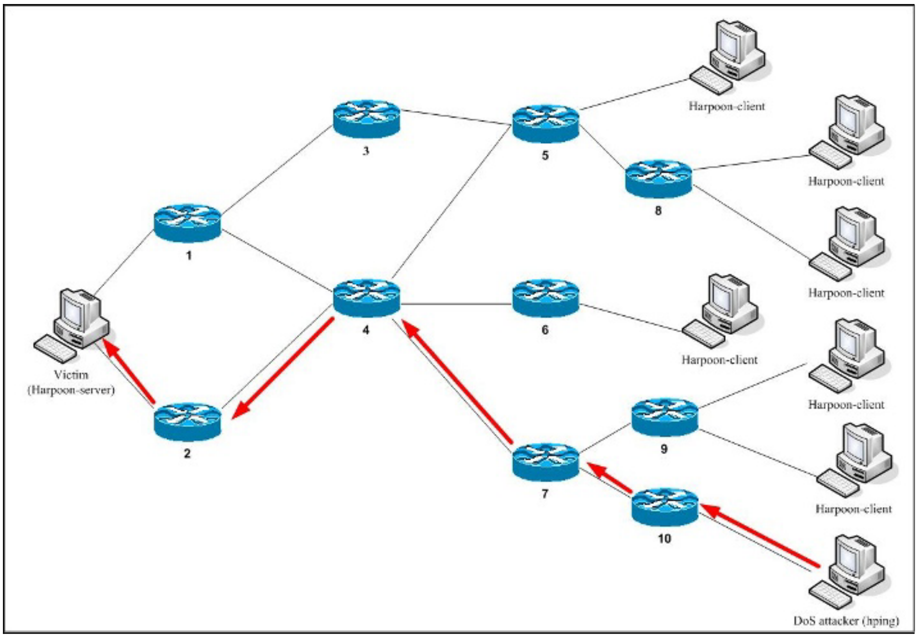


Fig. 5. DoS attack scenario

Once the DoS flows are identified, the flow management component can find out the octets sent by the DoS flows with the source address in the suspected source address list. The finding then will be fed to the traceback component to find the DoS attack path.

The results of the traceback are shown in the following figures presented in three dimensional graphs, where the x-axis represents the path discovered by ants, the y-axis represents the number of iterations, and the z-axis represents the number of ants in y-th cycle found x-th path. The attack path found by the proposed ant colony based traceback method is the one with the most ants.

Figure 6 shows the results of the traceback with full flow information provided by the network. The proposed traceback method explores all the possible attack paths in the initial stage of traceback and the ants would tend to converge to the attack path in the following iterations. After about half of the simulation, most ants will converge on the DoS attack path.

According to the results of the preliminary experiment, we verify that the proposed solution can find out the DoS attack path in case all the routers in the network provide flow information. However, in real environments, some flow information might be lost, especially at the router on the DoS attack path. Other experimental results are eliminated due to the length of the paper, but they all conclude that the proposed solution can find the DoS path efficiently and correctly.

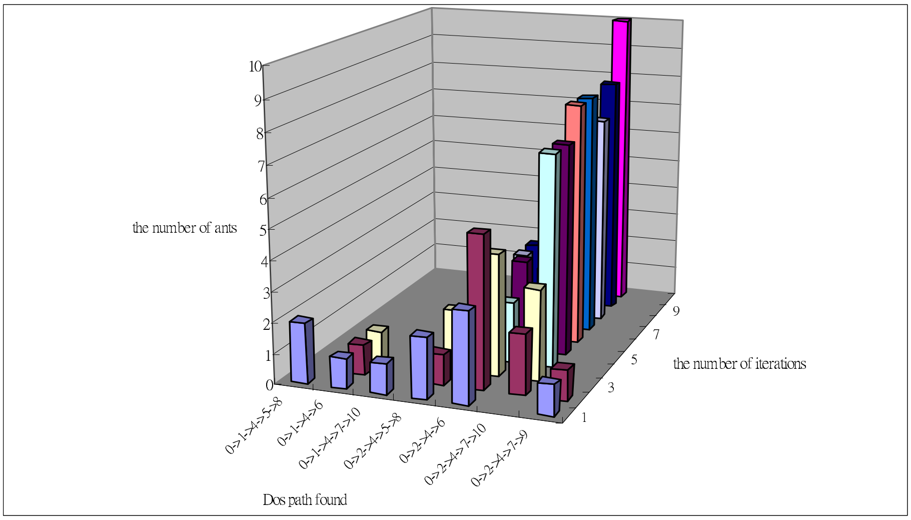


Fig. 6. The results of the traceback

5 Conclusion

DoS attacks have become one of the major threats in the Internet and cause massive revenue loss for many companies. However, DoS attacks are often associated with spoofed source addresses, making them hard to identify the attacker. A proactive approach to DoS attacks is to find the original machine which issues the attack and stop the malicious traffic.

In this research, the traceback based on ant colony is proposed to identify the DoS attack origin. Unlike the previous traceback schemes, such as packet marketing and logging, which use packet level information, the proposed traceback approach uses flow level information. Although the packet level information provides detailed information about the network, the high processing cost is a challenge for deploying those IP traceback methods in the real networks.

Ant colony algorithms have been successfully applied to various routing and optimization problems. Based on our observation, the proposed traceback problem is a variation of a routing problem and hence an ant colony based algorithm could be used to find the DoS attack path.

The proposed method is verified and evaluated through simulation. The simulation results show that the proposed method can successfully and efficiently find the DoS attack path in various simulated network environments. Hence, we conclude that the proposed solution is an efficient method to find the DoS attack origin in the networks.

The proposed DoS traceback method can identify the DoS attack path in case of the spoofed source addresses. However, there are other attacks with spoofed source addresses which need to be identified. Ant algorithms or other artificial intelligent approaches could be further investigated for more generalized IP traceback problems. A distributed flow management might be more scaleable for large networks. Further study on the practical implementation and deployment on a large network can be done to evaluate the scalability of the proposed solution.

References

1. Computer Security Institute, "CSI/FBI Computer Crime and Security Survey, "2003, http://www.crime-research.org/news/11.06.2004/423/.
2. S. Savage, D. Wetherall, A.Karlin, and T.Anderson ., "Network Support for IP Traceback," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, 2001, pp.226–237 .
3. D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. IEEE INFOCOM, IEEE CS Press,* 2001, pp. 878–886.
4. A.C. Soneren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tachakountio, B. Schwartz, S.T. Kent and W.T. Strayer ,"Single-packet IP Traceback," *IEEE/ACM Trans. Networking*, vol. 10, no.6, 2002, pp.721–734.
5. W.T Strayer, C.E. Jones, F. Tachakountio, B. Schwartz, R.C. Clements, M. Condell and C. Partridge ,"Traceback of Single IP Packets Using SPIE," *Proc. DARPA information Survivability Conference and Exposition* – vol. 2 April 22 -24, 2003 Washington, DC. pp. 266
6. G. Upton, "Swarm Intelligence," http://www.cs.earlham.edu/~uptongl/project/Swarm_Intelligence.html\.
7. M. Dorigo, V. Maniezzo & A. Colorni," The Ant System: An Autocatalytic Optimizing Process," Technical Report No. 91-016 Revised, Politecnico di Milano, Italy, 1991.
8. Y. Gong ,"Detecting Worms and Abnormal Activities with NetFlow," http://www. securityfocus. com/ infocus/1796
9. Scientific Linux https://www.scientificlinux.org/
10. flow-tools information http://www.splintered.net/sw/flow-tools/
11. Stanford Stream data manager http://www-db.stanford.edu/stream/
12. VMware http://www.vmware.com/
13. zebra http://www.zebra.org/
14. fprobe http://fprobe.sourceforge.net/
15. hping http://www.hping.org/

Optimisation of Constant Matrix Multiplication Operation Hardware Using a Genetic Algorithm

Andrew Kinane, Valentin Muresan, and Noel O'Connor

Centre for Digital Video Processing, Dublin City University, Dublin 9, Ireland
kinanea@eeng.dcu.ie

Abstract. The efficient design of multiplierless implementations of constant matrix multipliers is challenged by the huge solution search spaces even for small scale problems. Previous approaches tend to use hill-climbing algorithms risking sub-optimal results. The three-stage algorithm proposed in this paper partitions the global constant matrix multiplier into its constituent dot products, and all possible solutions are derived for each dot product in the first two stages. The third stage leverages the effective search capability of genetic programming to search for global solutions created by combining dot product partial solutions. A bonus feature of the algorithm is that the modelling is amenable to hardware acceleration. Another bonus feature is a search space reduction early exit mechanism, made possible by the way the algorithm is modelled. Results show an improvement on state of the art algorithms with future potential for even greater savings.

1 Introduction

Applications involving the multiplication of variable data by constant values are prevalent throughout signal processing. Some common tasks that involve these operations are Finite Impulse Response filters (FIRs), the Discrete Fourier Transform (DFT) and the Discrete Cosine Transform (DCT). Optimisation of these kinds of constant multiplications will significantly impact the performance of such tasks and the global system that uses them. The examples listed are instances of a more generalised problem – that of a linear transform involving a constant matrix multiplication (CMM). The problem is summarised as follows: substitute all multiplications by constants with a minimum number of shifts and additions/subtractions (we refer to both as 'additions') [1]. The optimisation criterion may be extended beyond adder count to include factors like routability, glitching etc. but is restricted to adder count in this paper.

2 Problem Statement

A CMM equation $\mathbf{y} = \mathbf{A}\mathbf{x}$ (where $\mathbf{y}, \mathbf{x}$ are N-point 1D data vectors and $\mathbf{A}$ is an $N \times N$ matrix of M-bit fixed-point constants) may be thought of as a collection of N dot products with each dot product y_i expressed as follows:

$$y_i = \sum_{j=0}^{N-1} a_{ij} x_j, \quad i = 0, \ldots, N-1. \tag{1}$$

Each constant may be represented in signed digit (SD) form:

$$a_{ij} = \sum_{k=0}^{M-1} b_{ijk} 2^k, \quad b_{ijk} \in \{\bar{1}, 0, 1\}, \quad \bar{1} \equiv -1. \tag{2}$$

Combining Eqns. 1 and 2 yields a multiplierless dot product implementation requiring only adders and shifters:

$$y_i = \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} b_{ijk} 2^k x_j, \quad i = 0, \ldots, N-1. \tag{3}$$

The goal is to find the optimal sub-expressions across all N dot products in Eqn. 3 that require fewest adder resources. As reviewed below, three properties can be used in the classification of approaches to this problem: SD permutation, pattern search strategy and problem subdivision.

SD Permutation. Consider that each of the $N \times N$ M-bit fixed point constants a_{ij} have a finite set of possible SD representations. For example with $M = 4$ the constant $(-3)_{10}$ can be represented as either $(00\bar{1}\bar{1})_2, (0\bar{1}01)_2, (\bar{1}101)_2, (0\bar{1}1\bar{1})_2$ or $(\bar{1}11\bar{1})_2$. To find the optimal number of adders, all SD representations of a_{ij} should be considered since for a CMM problem Canonic Signed Digit (CSD) representation is not guaranteed to be optimal (as shown in Section 5). The difficulty is that the solution space is very large [2], hence SD permutation has thus far been applied only to simpler problems [2, 3]. Potkonjak et. al. acknowledge the potential of SD permutation but choose a single SD representation for each a_{ij} using a greedy heuristic. Neither of the recent CMM-specific algorithms in the literature apply SD permutation [4, 5], but the algorithm proposed in this paper does apply it.

Pattern Search. The goal of pattern searching is to find the sub-expressions in the 3D bit matrix b_{ijk} resulting in fewest adders. Usually b_{ijk} is divided into N 2D slices along the i plane (i.e. taking each CMM dot product in isolation). Patterns are searched for in the 2D slices independently before combining the results for 3D. An example 2D slice is shown in Eqn. 4, a 4-point dot product with random 8-bit SD constants.

$$y_i = \underbrace{\begin{bmatrix} 0.9375 \\ 0.921875 \\ 0.6013625 \\ 0.1328125 \end{bmatrix}^T}_{a_{ij} \text{ at } \mathbf{A} \text{ row } i} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2^{-7} \\ 2^{-6} \\ 2^{-5} \\ 2^{-4} \\ 2^{-3} \\ 2^{-2} \\ 2^{-1} \\ 2^{0} \end{bmatrix}^T \underbrace{\begin{bmatrix} 0 & 0 & 1 & \bar{1} \\ 0 & \bar{1} & 0 & 0 \\ 0 & 0 & \bar{1} & 1 \\ \bar{1} & 1 & 0 & 0 \\ 0 & 1 & 1 & \bar{1} \\ 0 & \bar{1} & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}}_{\text{2D slice of } b_{ijk}} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4}$$

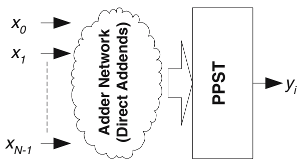


Fig. 1. P1D Architecture

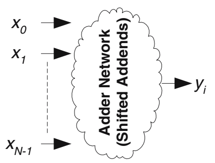


Fig. 2. P2D Architecture

Algorithms may search for horizontal/vertical patterns (P1D) or diagonal patterns (P2D) in the 2D slice. The P1D strategy implies a two-layer architecture of a network of adders (with no shifting of addends) to generate distributed weights for each row followed by a fast partial product summation tree (PPST) to carry out the shift accumulate (Fig. 1). The P2D strategy implies a one-layer architecture (Fig. 2) of a network of adders that in general may have shifted addends (essentially merging the two layers of the P1D strategy).

Potkonjak et. al. use the P1D strategy and search for horizontal patterns while others use the P2D strategy [4, 5]. However, these approaches select sub-expressions iteratively based on some heuristic criteria that may preclude an optimal realisation of the global problem. This is because the order of sub-expression elimination affects the results [6]. The proposed algorithm sidesteps this issue by building parallel solutions using the P1D strategy.

Problem Sub-Division. As in any hardware optimisation problem, synthesis issues should be considered when choosing sub-expressions for an N-point dot product (a 2D slice). If N is large (e.g. 1024-point FFT) then poor layout regularity may result from complex wiring of sub-expressions from taps large distances apart in the data vector. Indeed a recent paper has shown that choosing such sub-expressions can result in a speed reduction and greater power consumption [7]. It is therefore sensible to divide each N-point dot product into N/r sub r-point dot product chunks, where $r < N$ and $r \in \mathbb{Z}$, and optimise each chunk independently. The CMM problem hence becomes N/r independent sub problems, each with N dot products of length r (Fig. 3). The optimal choice of r is problem dependent, but the proposed algorithm currently uses $r = 4$ for reasons outlined subsequently. Eqn. 4 is an example of a sub dot product with $r = 4$.

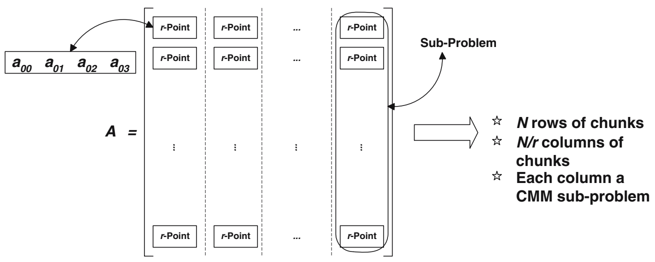

a_{00} a_{01} a_{02} a_{03}
A =
r-Point
...
Sub-Problem
☆ N rows of chunks
☆ N/r columns of chunks
☆ Each column a CMM sub-problem

Fig. 3. CMML Divide and Conquer

3 Proposed Efficient Modelling Solution

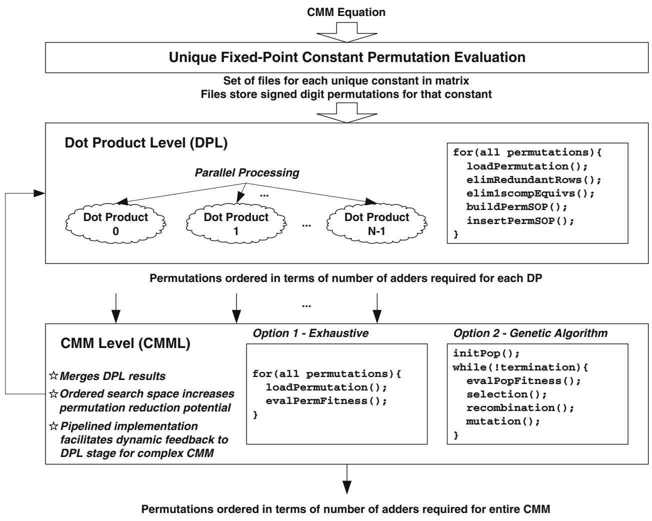
The CMM problem is a difficult discrete combinatorial problem and currently requires a shift to a higher class of algorithms for more robust near-optimal solutions. This is because the current approaches are greedy hill-climbing algorithms and the associated results are very problem dependent [6]. The challenge is in the modelling of the problem to make it amenable to efficient computation. The algorithm proposed here models the problem in such a way as to make it amenable to so-called near-optimal algorithms (genetic algorithms (GAs), simulated annealing, tabu-search) and also hardware acceleration. The proposed approach incorporates SD permutation of the matrix constants and avoids hill-climbing by evaluating parallel solutions for each permutation. Such an approach is computationally demanding but the algorithm has been modelled with this in mind and incorporates innovative fast search techniques to reduce this burden.

The proposed algorithm permutes the SD representations of the constants in $\mathbf{A}$. For each permutation, parallel solution options are built based on different sub-expression choices. These parallel implementations are expressed as a sum of products (SOP), where each product term in the SOP represents a particular solution (with an associated adder count). The SD permutation is done on each CMM dot product in isolation (Section 4.1), and the results are subsequently combined (Section 4.2). The algorithm searches for the combined SOP that represents the overall best (in terms of adder count) sub-expression configuration to implement the CMM equation. Previous approaches derive one implementation option (akin to a single term SOP) whereas the proposed approach derives parallel implementations (a multi-term SOP). It is this multi-term SOP approach and its manipulation (Section 4) that make the algorithm suitable for GAs and hardware acceleration.

The proposed algorithm currently uses the P1D strategy, so it searches for horizontal sub-expression patterns of $\{\pm 1\}$ digits in a 2D slice. The proposed SOP modelling idea can be extended to cover the P2D strategy by simply extending the digit set from $\{\pm 1\}$ to $\{\pm 1, \pm 2, \pm\frac{1}{2}, \pm 4, \pm\frac{1}{4}, \ldots\}$. To save space, the reasoning for this idea is not elaborated upon in this paper, but is targeted as future work.

4 The Proposed CMM Optimisation Algorithm

The proposed approach is a three stage algorithm as depicted in Fig. 4. Firstly all SD representations of the M-bit fixed point constants are evaluated using an M-bit radix-2 SD counter (digit set $\{\bar{1}, 0, 1\}$). Then, each dot product in the CMM is processed independently by the dot product level (DPL) algorithm. Finally the DPL results are merged by the CMM level (CMML) algorithm. The three steps may execute in a pipelined manner with dynamic feedback between stages. This offers search space reduction potential as outlined subsequently.


CMM Equation
Unique Fixed-Point Constant Permutation Evaluation
Set of files for each unique constant in matrix
Files store signed digit permutations for that constant
Dot Product Level (DPL)
Parallel Processing
Dot Product 0
Dot Product 1
Dot Product N-1
Permutations ordered in terms of number of adders required for each DP
CMM Level (CMML)
☆Merges DPL results
☆Ordered search space increases permutation reduction potential
☆Pipelined implementation facilitates dynamic feedback to DPL stage for complex CMM
Option 1 - Exhaustive
Option 2 - Genetic Algorithm
Permutations ordered in terms of number of adders required for entire CMM

```
for(all permutations){
  loadPermutation();
  elimRedundantRows();
  elim1scompEquivs();
  buildPermSOP();
  insertPermSOP();
}
```

```
for(all permutations){
  loadPermutation();
  evalPermFitness();
}
```

```
initPop();
while(!termination){
  evalPopFitness();
  selection();
  recombination();
  mutation();
}
```

Fig. 4. Summary of the CMM Optimisation Algorithm

4.1 Dot Product Level (DPL) Stage

The DPL algorithm iteratively builds a SOP, and the final SOP terms are the unique sub-expression selection options after considering all SD permutations of the dot product constants in question. The final SOP terms are listed in increasing order of the number of adders required by the underlying sub-expressions.

Each SOP term is represented internally as a data structure with elements `p_vec` (a bit vector where each set bit represents a specific adder to be resource allocated) and `hw` (the Hamming weight of `p_vec` that records the total adder requirement). The number of possible two input additions is equivalent to the combinatorial problem of leaf-labelled complete rooted binary trees [8]. With $r = 4$, the number of possibilities is 180 (proof omitted to save space) and the general series in r increases quickly for $r > 4$. We are currently researching an automated method for configuring the DPL algorithm for any r. Currently, however, each `p_vec` is a 180-bit vector with a `hw` equal to the number of required adders.

The DPL algorithm executes for each SD permutation of the dot product constants in question, and builds a 'permutation SOP' at each iteration. This process is described in detail in [9]. The permutation SOP for Eqn. 4 is given by Eqn. 5 where p_v means bit v is set in the 180-bit `p_vec` for that SOP term.

$$\begin{aligned}&((p_{11})(p_6)(p_3)(p_{51})(p_{10})(p_0))\text{ OR}\\&((p_{11})(p_6)(p_{10})(p_{52})(p_0))\text{ OR}\\&((p_{11})(p_6)(p_{53})(p_{10})(p_0))\end{aligned} \tag{5}$$

The first term in Eqn. 5 has `hw` = 6 so it requires 6 unique additions (+PPST) to implement Eqn. 4 whereas the latter two options only require 5 unique additions (+PPST). Obviously one of the latter two options is more efficient if

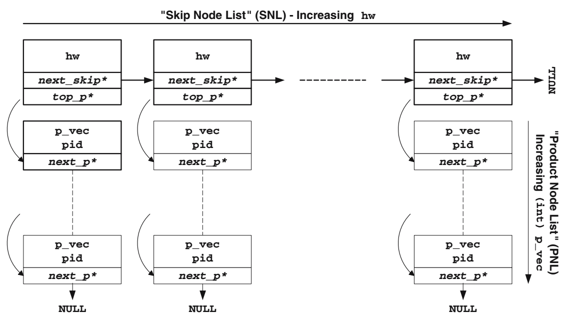

"Skip Node List" (SNL) - Increasing hw
hw
next_skip*
top_p*
p_vec
pid
next_p*
NULL
"Product Node List" (PNL) Increasing (int) p_vec

Fig. 5. DPL Skip List Arrangement

implementing this dot product in isolation. However, when targeting a CMM problem one must consider the CMM level, and it may be that permuting the first option at CMML gives a better overall result since it may overlap better with requirements for the other dot products. Hence it is necessary to store the entire SOP for each permutation at DPL and then permute these at CMML to get the guaranteed optimal.

The algorithm checks each term in the permutation SOP produced at each DPL iteration to see if it has already been found with a previous permutation. If so it is discarded – only unique implementations are added to the global list. This global list is implemented using a 2D skip list to minimise the overhead of searching it with a new term from the current permutation SOP (Fig. 5) [9]. In the horizontal direction there are 'skip nodes' ordered from left to right in order of increasing `hw` in the skip node list (SNL). In the vertical direction there are 'product nodes' and each skip node points to a product node list (PNL) of ordered product nodes where each product node in the PNL has the same number of bits set (i.e. `hw`) in its `p_vec` bit vector. When inserting a new term into the list, a unique permutation ID (`pid`) is added to the node along with `p_vec` so that the SD permutation that generated it can be reconstructed.

The DPL algorithm is dominated by low level operations such as comparisons, Boolean logic and bit counting. Indeed profiling shows that on average 60% of the computation time is consumed by bit counting (50%) and bitwise OR (10%). Such tasks can readily be accelerated in hardware by mapping the multi-term SOP to a FIFO structure and the logic OR operations to OR gates.

4.2 Constant Matrix Multiplication Level (CMML) Stage

Once the DPL algorithm has run for each of the dot products in the CMM, there will be N 2D skip lists – one for each of the N dot products examined. The task now is to find the best set of overlapping product nodes for all of the CMM dot products, with one node for each dot product. Overlapping nodes have similar `p_vec` set bits, and this results in adder resource sharing when implementing the CMM. It is expected (though not guaranteed) that since the skip lists are ordered with the lowest `hw` PNL first, the optimal result will be converged upon quickly saving needless searching of large areas of the permutation space. The

CMML algorithm searches for the optimal overlapping nodes from each of the DPL lists.

Exhaustive Approach. An exhaustive CMML algorithm permutes the terms in each skip list with terms from others, starting from the top of each. For each permutation, N product nodes (one from each list) are combined using bitwise OR and bit counting similar to the techniques used in the DPL algorithm. The value of `hw` of the combined node represents the number of adders necessary to implement the CMM for the current permutation. The potential exists to use the lowest `hw` value found thus far to rule out areas of the search space – hence the early exit mechanism referred to previously. For example if an improved value of `hw` = 5 is found for a CMML solution, there is no point in searching DPL PNLs with `hw` > 5 since they are guaranteed not to overlap with other DPL PNLs and give a better result than 5. The current best value of `hw` at CMML level could also be fed back to the DPL algorithm to reduce the size of the skip lists generated by DPL (and hence permutation space) without compromising optimality. However, despite the DPL skip list ordering, the huge permutation space means that the exhaustive CMML approach is not tractable, especially as N increases.

Genetic Programming Approach. The proposed modelling of the CMM problem and bit vector representation of candidate solutions means that the CMML algorithm is very amenable to GAs. The bit vectors can be interpreted as chromosomes and the value of `hw` can be used to build an empirical fitness function (the less adders required the fitter the candidate). A proposed GA to implement the CMML algorithm is summarised in Algorithm 1.

Algorithm 1: GA-based CMML Algorithm

```
init_pop();
while !termination condition do
    eval_pop_fitness();
    selection();
    recombination();
    mutation();
end
```

A candidate solution `c` is represented by a set of N pointers `slp[i][c]`, where each pointer addresses a product node in dot product skip list i ($i = 0, 1, \ldots, N-1$). The N product nodes are combined using bitwise OR and bit counting as described in [9]. The task of the GA is to find the DPL component product nodes that overlap as much as possible resulting in the fewest adders necessary to implement the CMM with a P1D architecture (Fig. 1). The individual steps of Algorithm 1 are described in the following sections.

Step 0 – Initialise Population. The size of the population is determined by the parameter *pop_size*. Since the DPL stage results are ordered as described in Section 4.1, the population is initialised with candidates (sets of pointers) near the top of the DPL lists. This is achieved by weighting the selection of the initial candidates. Let z represent the address each of the N component pointers `slp[i][c]` can assume for any candidate `c`. For each pointer, z is in the range $0 \leq z \leq \mathrm{NP}_i$, where NP_i is the number of product nodes in skip list `i`. The algorithm randomly sets the pointer address z for all N pointers for each of the initial *pop_size* candidates according to an exponential probability mass function Eqn. 6.

$$p(z) = \frac{1}{\mu} \exp(-z/\mu) \tag{6}$$

According to Eqn. 6, the lower the value of parameter μ, the more likely a candidate is to have DPL component pointers nearer the top of the respective DPL skip lists (i.e. z tends to zero for each of the N pointers).

Step 1 – Population Fitness Evaluation. The fitness of a candidate solution is obtained by doing a bitwise OR of all of the component pointees followed by bit counting. The lower the resultant bit count the better, as it means less adder resources are required to implement the CMM problem with a P1D hardware architecture. In future work we intend extending the fitness function to include factors like fanout and logic depth, e.g. Eqn. 7. Currently Eqn. 7 is restricted to adder count only.

$$f = \alpha(\text{Adder Count}) + \beta(\text{Fanout}) + \gamma(\text{Logic Depth}) + \ldots \tag{7}$$

Step 2 – Selection. A good selection method should maintain an appropriate balance between selective pressure and population diversity. The proposed method is a variation of Goldberg's Boltzmann Tournament Selection algorithm [10]. Tournament selection involves a pure random selection of t individuals ($t \leq pop\_size$) that compete in terms of fitness against each other and the winner is selected. This process is repeated pop_size times. However, we propose to use a strategy with a 'fuzzy' selection decision with $t = 2$. Goldberg's algorithm is based upon simulated annealing, i.e. at high 'temperatures' there is a greater chance that weak candidates may be selected, which enhances population diversity and makes it less likely that the algorithm will get stuck in local optima. As the temperature cools, the strong candidates begin to dominate selection since the algorithm should be converging on the true optimum.

The proposed approach uses Eqn. 8 which is plotted along with the exponent of $X = \frac{f(j)-f(k)}{T}$ in Fig. 6 where $f(j)$ and $f(k)$ are the fitness values of candidates j and k respectively.

$$W = \frac{1}{1 + e^{\frac{f(j)-f(k)}{T}}} = \frac{1}{1 + e^{X}} \tag{8}$$

As is clear from Fig. 6, as the temperature T decreases, the value of the exponential term X moves further from the central vertical axis for a fixed $f(j)$ and $f(k)$. As T decreases $W \to 1$ when $f(j) < f(k)$ and $W \to 0$ when $f(k) < f(j)$.

The original Boltzmann tournament selection algorithm proposed by Goldberg uses $t = 3$, and lets W equal the probability that j wins the tournament and

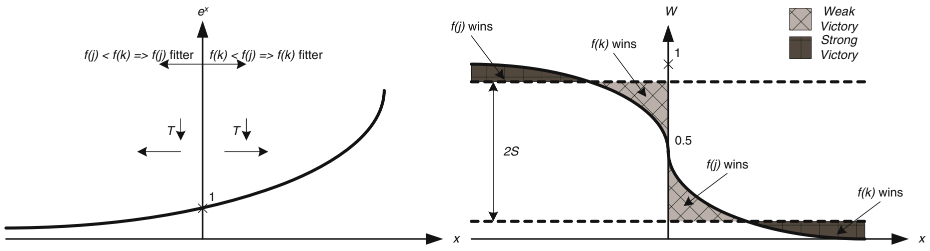


Fig. 6. Boltzmann Decision Based Simulated Annealing

($1-W$) be the probability that k wins the tournament [10]. We propose a variation on Goldberg's algorithm by introducing a fuzzy select threshold S to enhance the population diversity. Using S, the selection algorithm can be programmed to have a higher probability of selecting a weak candidate as a tournament victor when the temperature T is high in the early generations. As the temperature decreases and the algorithm converges on the optimum, the stronger candidate has a greater chance of victory. The approach is summarised in Algorithm 2.

Algorithm 2: Fuzzy Boltzmann Tournament Selection Algorithm

```
if $f(j) < f(k)$ then
    if $W >$ (0.5 + S) then $j$ wins (strong victory);
    else $k$ Wins (weak victory)
end
else if $f(j) > f(k)$ then
    if $W <$ (0.5 - S) then $k$ wins (strong victory);
    else $j$ Wins (weak victory)
end
else
    Choose pure random winner
end
```

To summarise, the proposed selection method maintains a balance between population diversity and selection strength. The selection decision depends on the relative fitness of competing individuals, the temperature T and the fuzzy select threshold S. Since the GA should converge on globally optimal solutions as the generations iterate, the parameters T and S should decay over the generations to select the strong candidates with higher probability.

Step 3 – Recombination. After pop_size individuals have been selected, a proportion of these are further selected for uniform crossover based on a probability p_c. Since each candidate is represented by N pointers, the uniform crossover process generates a random N-bit binary mask. Each bit location in the mask determines the mixture of genetic material from the parents each offspring is created with. Consider Fig. 7. If a bit location in the mask is '0', the corresponding pointer component for offspring '0' is created respectively from parent '0', and

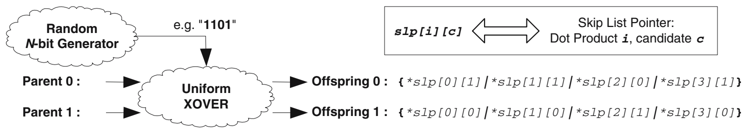

Random
N-bit Generator
e.g. "1101"
Parent 0 :
Parent 1 :
Uniform
XOVER
Offspring 0 : {*slp[0][1]|*slp[1][1]|*slp[2][0]|*slp[3][1]}
Offspring 1 : {*slp[0][0]|*slp[1][0]|*slp[2][1]|*slp[3][0]}
slp[i][c]
Skip List Pointer:
Dot Product i, candidate c

Fig. 7. Uniform Crossover Example

the corresponding component for offspring '1' is created from parent '1'. The opposite creation process occurs if the bit is '1'.

Step 4 – Mutation. After selection and crossover, the DPL component pointers of each candidate undergo mutation based on a probability p_{mut}. If mutation is applied, the degree of mutation is determined by a value M, where $M \in \mathbb{Z}$. A pointer selected for mutation moves M pointer locations up ($M < 0$) or down ($M > 0$) its associated DPL skip list. The range of mutations possible depends on the value of a parameter M_{max}. The value for M is determined based on a binomial probability density function $p(M)$ Eqn. 9. This distribution means that if mutation is applied, smaller mutations are more likely than large mutations.

$$p(M) = \frac{(2M_{max} - 1)!}{M!((2M_{max} - 1) - M)!} 0.5^M (0.5)^{((2M_{max}-1)-M)} \tag{9}$$

To allow positive or negative mutations, the binomial distribution is re-aligned about $M = 0$ (where $p(0) = 0$ because $M = 0$ means no mutation).

After this step, the new population forming the next generation is ready and the process loops back to step 1. The process continues iterating steps 1-4 until a termination condition is met (a fixed number of generations or a time constraint).

4.3 Genetic Algorithm Parameter Selection

Choosing values for the parameters that steer a GA is a difficult problem in itself. The parameter values in Table 1 have been obtained empirically by trial and error, and future work will investigate a more sophisticated method. Based on empirical observations, the tuned parameter values in Table 1 imply that the CMML GA produces better results when there is weak selective pressure (strong diversity). The reason for this is likely to be because the variance of the solution

Table 1. CMML Genetic Algorithm Parameters

Parameter	Name	Value
pop_size	Population Size	3000
μ	Initialisation Weight	10.0
T	Selection Temperature	0.001
S	Selection Threshold	0.4
p_c	Crossover Probability	0.98
p_{mut}	Mutation Probability	0.08
M_{max}	Max Mutation Size	6

space fitness values is quite low, according to the current fitness function, relative to the size of the solution space. Hence the current search is almost a "needle in a haystack" search, so a healthy diversity is needed. Future work on this algorithm aims to increase the dimensionality of the fitness function to include other factors like logic depth and fanout as well as adder count. Extending the fitness function should increase the granularity of the fitness values in the solution space. Hence the tuned genetic algorithm parameters are likely to change in future so that the selective pressure will increase.

5 Experimental Results

For a fair comparison with other approaches, the number of 1-bit full adders (FAs) allocated in each optimised architecture should be used as opposed to 'adder units', since the bitwidth for each unit is unspecified in other publications apart from in [5]. FA count more accurately represents circuit area requirements. Using the 8-point 1D DCT ($N = 8$ with various M) as a benchmarking CMM problem, Table 2 compares results with other approaches based on adder units and FAs where possible. Our approach compares favourably with [5] in terms of FAs (see FA% savings in Table 2), even though this gain is not reflected by the number of adder units required.

Our previous results were based on running the proposed CMML GA with untuned parameters for 100000 generations [9]. Using the tuned parameters of Table 1, our results clearly improve as is evident from Table 2. The tuned parameters also find these improved solutions after fewer generations (1000). For each of the benchmarks in Table 2, the tuned parameters cause the proposed algorithm to invoke its search space reduction mechanism (Section 4.2). This reduces the search space from the order of 10^{20} to 10^{17} without compromising the quality of the results , representing a reduction of more than 99%. The hypothesis of achieving extra saving by permuting the SD representations is validated by the fact that the best SD permutation corresponding to our results in Table 2 are not the CSD permutation.

Even given the savings illustrated in Table 2, there exists significant potential for improvement:

1. Investigation of an optimal value for r, that is the optimal sub division of large CMM problems into independent chunks. This can only be truly

Table 2. 1D 8-point DCT Adder Unit / Full Adder Requirements

CMM	Initial	[1]	[4]	[5]		Ours					
						Untuned GA [9]			Tuned GA		
	+	+	+	+	FA	+	FA	FA%	+	FA	FA%
DCT 8bit	300	94	65	56	739	78	730	1.2	77	712	3.7
DCT 12bit	368	100	76	70	1202	109	1056	12.1	108	1048	12.8
DCT 16bit	521	129	94	89	2009	150	1482	26.2	141	1290	35.8

evaluated if synthesis parameters such as fanout and routability are included in the fitness function as well as FA count.

2. The integration of the P2D strategy mentioned earlier. It is likely that there exists an upper bound on the number of rows apart within the b_{ijk} slice between which useful sub-expressions will be found. This is because if sub-expression addends come from rows far apart in b_{ijk}, the adders inferred have a large bitwidth.
3. Extension of the fitness function as indicated, and subsequent tuning of the GA parameters.

6 Conclusions

The general multiplierless CMM design problem has a huge search space, especially if different SD representations of the matrix constants are considered. The proposed algorithm addresses this by organising the search space effectively, and by using a GA to quickly search for near optimal solutions. Experimental results validate the approach, and show an improvement on the current state of the art.

References

1. Potkonjak, M., Srivastava, M.B., Chandrakasan, A.P.: Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination. IEEE Transactions on Computer-Aided Design of Integrated Circuits **15** (1996) 151–165
2. Dempster, A.G., Macleod, M.D.: Digital Filter Design Using Subexpression Elimination and all Signed-Digit Representations. In: Proc. IEEE International Symposium on Circuits and Systems. Volume 3. (2004) 169–172
3. Dempster, A.G., Macleod, M.D.: Using all Signed-Digit Representations to Design Single Integer Multipliers using Subexpression Elimination. In: Proc. IEEE International Symposium on Circuits and Systems. Volume 3. (2004) 165–168
4. Macleod, M.D., Dempster, A.G.: Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers. IEE Electronics Letters **40** (2004) 651–652
5. Boullis, N., Tisserand, A.: Some Optimizations of Hardware Multiplication by Constant Matrices. IEEE Transactions on Computers **54** (2005) 1271–1282
6. Macleod, M.D., Dempster, A.G.: Multiplierless FIR Filter Design Algorithms. IEEE Signal Processing Letters **12** (2005) 186–189
7. Martinez-Peiro, M., Boemo, E.I., Wanhammar, L.: Design of High-Speed Multiplierless Filters Using a Nonrecursive Signed Common Subexpression Algorithm. IEEE Transations on Circuits and Systems II **49** (2002) 196–203
8. Andres, S.D.: On the number of bracket structures of n-operand operations constructed by binary operations (2005) private communication.
9. Kinane, A., Muresan, V., O'Connor, N.: Towards an Optimised VLSI Design Algorithm for the Constant Matrix Multiplication Problem. In: Proc. IEEE International Symposium on Circuits and Systems, Kos, Greece (2006)
10. Goldberg, D.: A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. Complex Systems **4** (1990) 445–460

Finding Compact BDDs Using Genetic Programming

Ulrich Kühne and Nicole Drechsler

Institute of Computer Science, University of Bremen, Bremen 28359, Germany
{ulrichk, nd}@informatik.uni-bremen.de

Abstract. Binary Decision Diagrams (BDDs) can be used to design multiplexor based circuits. Unfortunately, the most commonly used kind of BDDs – ordered BDDs – has exponential size in the number of variables for many functions. In some cases, more general forms of BDDs are more compact. In constrast to the minimization of OBDDs, which is well understood, there are no heuristics for the construction of compact BDDs up to today. In this paper we show that compact BDDs can be constructed using Genetic Programming.

1 Introduction

Decision Diagrams (DDs) are used for the representation of Boolean functions in many applications of VLSI CAD, e.g. in the area of logic synthesis [1] and verification [2]. In the meantime DD-based approaches have also been integrated in commercial tools.

The state-of-the-art data structure are *Ordered Binary Decision Diagrams* (OBDDs) [2]. Since OBDDs are often not able to represent Boolean functions efficiently due to the ordering restriction [3],[4],[5] many researchers have extended the OBDD concept mainly in two directions:

1. Consider different decomposition, e.g. *ordered functional decision diagrams* (OFDDs) [6] and *ordered Kronecker functional decision diagrams* (OKFDDs) [7] make use of AND/EXOR based decompositions.
2. Loosen the ordering restriction, e.g. *general* BDDs (GBDDs) [8] allow variables to occur several times.

Following the second approach, of course, the most powerful technique is to have no restriction on the ordering at all, i.e. to use BDDs without any restrictions on ordering or variable repetition. BDDs are often exponentially more succinct than OBDDs and also for the applications mentioned above the ordering restrictions are often not needed. The main reason why OBDDs have been used more frequently is that efficient minimization procedures exist, like e.g. sifting [9]. For BDDs similar techniques are not available.

Evolutionary approaches have also been applied successfully to OBDDs, but there the problem reduces to finding a good variable ordering, i.e. a permutation of the input variables [10]. In [11] Genetic Programming has been applied to a tree-like form of BDDs with some additional constraints.

In this paper we present an approach to BDD minimization based on Genetic Programming. In contrast to the minimization of OBDDs we carry out all operations directly on the graph structure of the BDD. By this, we present the first heuristic approach to BDD minimization. Experimental results are given to demonstrate the efficiency of the approach.

2 Preliminaries

2.1 Binary Decision Diagrams

A BDD is a directed acyclic and rooted graph $G_f = (V, E)$ representing a Boolean Function $f : \mathbb{B}^n \longrightarrow \mathbb{B}^m$. Each internal node v is labeled with a Variable $label(v) = x_i \in X_n = \{x_1, \ldots, x_n\}$, and has two successors $then(v)$ and $else(v)$. The terminal nodes are labeled with 1 or 0 corresponding to the constant Boolean functions. In each internal node the Shannon decomposition $g = x_i g|_{x_i=1} + \overline{x}_i g|_{x_i=0}$ is carried out. The m root nodes represent the respective output functions.

By restricting the structure of the BDD, special classes of BDDs can be derived:

– A BDD is *complete*, if on each path from a root to a terminal node each variable is encountered exactly once.
– A BDD is *free* (FBDD), if each variable is encountered at most once on each path from a root to a terminal node.
– A BDD is *ordered* (OBDD), if it is free and the variables appear in the same order on each path from a root to a terminal node.

OBDDs are a widely used data structure in hardware design and verification because they are a canonical representation of Boolean Functions and they provide efficient synthesis algorithms. However, for many functions the size of the OBDD depends on the variable ordering. It may vary between linear and exponential in the number of variables [2]. A lot of research has focused on the so-called variable ordering problem, which is NP-hard [12]. Furthermore, there are functions for which all variable orderings lead to an OBDD with exponential size. In turn, for some of these functions there exist FBDDs or BDDs of polynomial size [13]. This means that releasing the read-once restriction and the ordering of variables can be advantageous. But in constrast to the minimization of OBDDs by finding a good or optimal variable ordering – which is well understood [14],[9] – there are no heuristics for the construction of small BDDs up to today.

2.2 BDD Circuits

BDDs can be directly mapped to a circuit based on multiplexors. If realized with *pass transistor logic*, multiplexor cells can be used for synthesis at low cost [15],[16],[1]. In the mapping, each internal node v of the BDD is replaced by a MUX cell. Then the 1-input (the 0-input) is connected to the MUX cell correspondig to $then(v)$ ($else(v)$). The select line is connected to the primary input

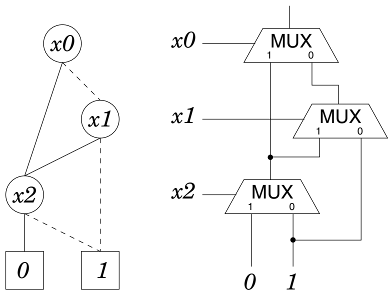


Fig. 1. Example for a BDD circuit

index(v). An example for the transformation is shown in Figure 1[1]. Obviously, the size of the BDD has direct influence on the chip area of the derived BDD circuit. For this reason, it is important to find a BDD representation as small as possible to minimize chip area.

3 Evolutionary Algorithm

The approach presented in this paper is based on Genetic Programming (GP) [17]. Originally, GP was used to evolve LISP programs. The method at hand does not consider programs, but works directly on the graph structure of the BDDs. Several operators are provided to customize the algorithm for a specific problem. The algorithm has been implemented on top of the *evolving objects* library [18], an open source C++ library for evolutionary algorithms. The target function is kept in memory as OBDD. For this the BDD package CUDD [19] is used. The aim of the evolutionary algorithm is formulated as follows:

> The objective is to evolve BDDs that are a *correct* and *compact* representation of a given target function.

3.1 Flow of the Algorithm

The general structure of the evolutionary algorithm is based on the cycle given by the EO library. The flow is depicted in Figure 2. The algorithm is parameterized via command line switches in most of its parts. All of the operators described below can be selected and the ratios can be adjusted individually. Furthermore the algorithm provides different methods for selection, replacement and different termination criterions. This high flexibility should enable the user to customize the flow according to the respective problem.

3.2 Representation

The individuals are directed acyclic graphs with multiple root nodes, each corresponding to an output of the represented function. By adopting some popular

[1] In the figures, solid lines denote *then* edges and dashed lines denote *else* edges.

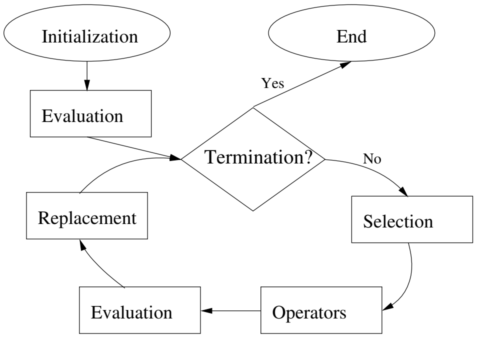


Fig. 2. The flow of the algorithm

techniques used in BDD packages, the graphs are always reduced, i.e. isomorphic subgraphs exist only once and there are no nodes with $then(v)$ and $else(v)$ being identical.

3.3 Evaluation Function

As mentioned above, there are two objectives – correctness and compactness. The two dimensions of fitness are ordered lexicographically, with correctness being the more important one. Initially there is no limitation for the structure of the represented BDDs. This means that there will be many individuals that do not represent the target function correctly. Such invalid individuals are given a worse fitness than the correct ones.

For the first dimension – correctness – it would be easy to test, if an individual represents the target function correctly or not. But this would draw no distinction between individuals that are "almost correct" and totally degenerated individuals. For this reason, a more sophisticated measure of correctness is used. The evaluation function calculates the ratio of assignments $a \in \mathbb{B}^n$ for which the function represented by the individual evaluates equivalently to the target function, i.e. the *correlation* between the individual's function and the target function. The computation of the correlation is realised by computing the XOR-BDD of the target function and the individual's function and then computing its ratio of satisfying assignments. For the latter step, the underlying BDD package provides an efficient implementation without considering each of the 2^n possible assignments.

The second dimension of fitness is the sum of the number of internal nodes of the individual and of the XOR-BDD already computed in the previous step of evaluation. The XOR-BDD is considered again in order to punish degenerated individuals that have few minterms in common with the target function. For a correct individual, the XOR-BDD is the zero function, and only the individual's nodes are counted.

3.4 Evolutionary Operators

Table 1 gives an overview of the genetic operators. Beyond the standard classes – initialization, recombination and mutation – there are some special operators, *functionally conserving* ones, which do not change the functional semantics of the individuals but the structure of the graphs.

Table 1. Genetic operators

initialization	`Init` `CuddInit`
recombination	`NodeXover` `OutputXover`

mutation	`VariableMut`
	`EdgeSwapMut`
	`EraseNodeMut`
	`AddMinterm`

functionally conserving	`Restructuring`
	`VariableDuplication`
	`CombinedRestructuring`
	`TautoReduction`
	`SplitReduction`
	`Resize`

Initialization. There are two types of initialization, `Init` and `CuddInit`. The first one generates random graphs with a given maximum depth. The second one creates OBDDs with a randomized variable ordering which represent the target function correctly. The name is derived from the underlying BDD package which is used for the synthesis of the OBDDs.

Crossover. `NodeXover` is basically a standard GP crossover, i.e. a node is selected from each parent, and the corresponding subgraphs are exchanged. As the individuals are rather DAGs than trees, it has to be assured that no cyclic subgraphs appear during the operation. The second crossover – `OutputXover` – performs a uniform crossover on the output nodes of the parent individuals. Thus it can only be applied to functions with multiple outputs. `OutputXover` is supposed to combine individuals that already provide good solutions for single output functions.

Mutation. Among the mutation operators there are three simple ones and the customized `AddMinterm` operator. `VariableMut` exchanges the variable of one randomly selected node. `EdgeSwapMut` selects one node and swaps its *then* and *else* egdes. `EraseNodeMut` removes one node from the graph by replacing it with one of its successors. This may be useful to eliminate redundant nodes. `AddMinterm` works as follows: first an assignment $a \in \mathbb{B}^n$ is generated. If the

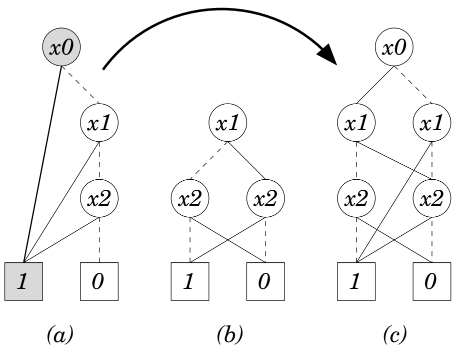


Fig. 3. Example for the `AddMinterm` operator

individual and the target function evaluate to different values under this assignment, a OBDD-like subgraph is added to the individual so that it will evaluate to the correct value afterwards. In order to create the subgraph, the target function is restricted in all variables that have been read on the path in the individual that corresponds to the assignment a. Then the new subgraph is appended to the end of this path. The operator can be used to speed up the algorithm, if the target function is relatively complex and so it would take too long to find a correct solution at all. It is a drawback that always OBDD-like subgraphs are created. This may lead to local optima that are hard to escape from.

Example 1. Figure 3 shows an example for the `AddMinterm` operator. Let the target function be the 3 bit parity function given by $f(x_0, x_1, x_2) = x_0 \oplus x_1 \oplus x_2$. Consider the individual in Figure 3(a) and the randomly chosen assignment $a = x_0\, x_1\, \overline{x_2}$. The corresponding path is highlighted in the figure. As on this path only x_0 is evaluated, the remaining function to be added is $f_{rest} = f|_{x_0=1} = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$. The OBDD for this function (see Figure 3(b)) is added to the end of the path, obtaining the new indiviual in Figure 3(c). Note that the correlation has increased from 5/8 to 7/8, i.e. only one of eight minterms is wrong after the application of `AddMinterm`.

Functionally Conserving Operators. Among the functionally conserving operators there are two operators that perform a local restructuring of the graphs. `Restructuring` searches for subgraphs that are isomorphic to one of the graphs shown in Figure 4 on the right and on the left. Then this subgraph is reduced to the one shown in Figure 4 in the middle. Note that the three graphs are functionally equivalent. The operator `VariableDuplication` duplicates a variable on a randomly selected path. The transformation is shown in Figure 5. In this way redundancy is added to the graph. In some cases this can lead to better solutions if the redundancy is removed elsewhere in the graph (this can be done by `TautoReduction` or `SplitReduction` which are described below). Furthermore the operator increases the diversity of the population. `CombinedRestructuring` is a combination of the two operators described above. Both are applied several times in random order.

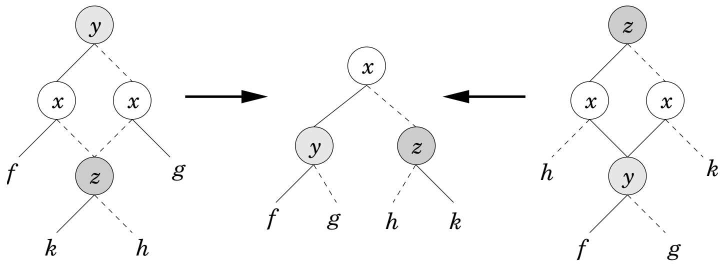


Fig. 4. The `Restructuring` operator

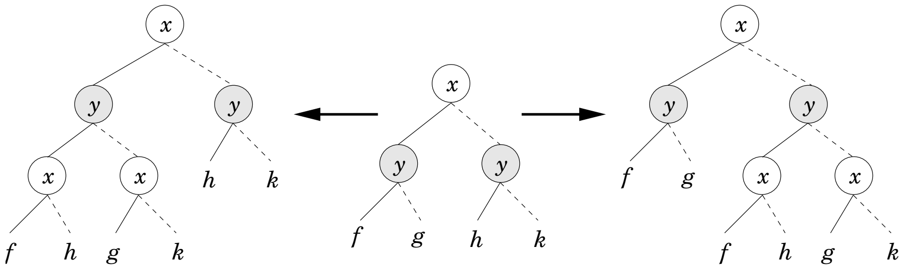


Fig. 5. The `VariableDuplication` operator

```
$curr\_lvl = TOP\_LEVEL$;
$done = false$;
while ($\neg done$) do
    for ($i = TOP\_LEVEL$ downto $curr\_lvl$) do
        for (each node in level $i$) do
            update path infos of child nodes;
        od
    od
    for (each node in level $curr\_lvl + 1$) do
        if node is redundant then remove node;
    od
    $curr\_lvl = curr\_lvl - 1$;
    if $curr\_lvl \leq 1$ then $done = true$;
od
```

Fig. 6. Pseudocode for `TautoReduction`

Finally there are two operators that try to reduce the number of nodes without changing the function of an individual. Algorithmically they are very similar. Figure 6 shows the `TautoReduction` operator in pseudocode. It searches for redundant nodes from top to bottom. A node v is identified redundant, if the variable $index(v)$ has already been evaluated to the same value on all paths reaching v. A redundant node can then be replaced by the appropriate successor.

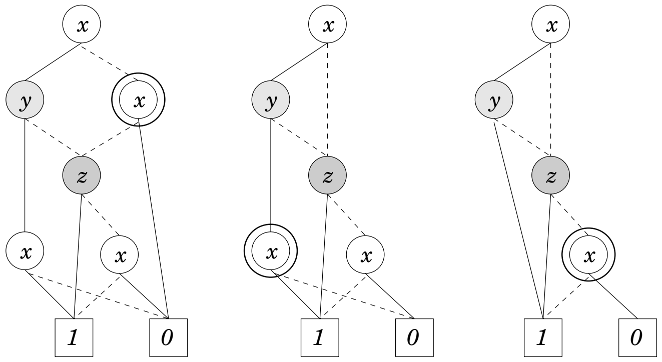


Fig. 7. Example for the `TautoReduction` operator

Example 2. Consider the individual shown in Figure 7. The marked node is redundant because x has already been evaluated to 0 above and it can be replaced by its *else* successor. In the next step, another node is identified redundant. In this case x has been evaluated to 1 before and the node is replaced by its *then* successor. The node marked in the third graph may not be removed because it can be reached on two paths on which x is evaluated to different values.

`SplitReduction` works similar, except that it does not remove redundant nodes but redirects edges instead. If in a node v the variable of one of its successor nodes has *always* been evaluated to a certain value and *never* to its complement, the edge to this node can be redirected to the appropriate successor node.

Example 3. Consider the individual in Figure 8. The *else* edge of the marked node can be redirected to the terminal 1-node. Furthermore the *then* edge can be redirected to the z node below because x has always been evaluated to 1 before.

By redirecting edges, `SplitReduction` will possibly "relieve" a node of redundant edges and make it in turn a candidate for `TautoReduction`. It could be

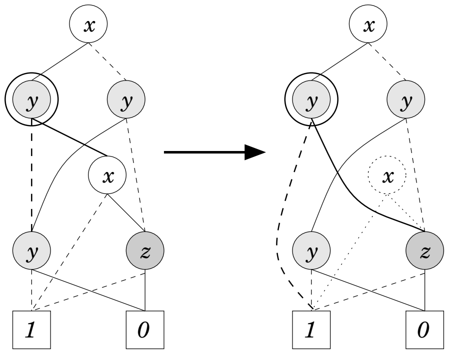


Fig. 8. Example for the `SplitReduction` operator

observed that the two operators work together quite well. Thus, there is a combination of these two operators, called `Resize`. `Resize` can be called at the end of every generation. It tries to reduce all individuals of the population using `TautoReduction` and `SplitReduction`, so that the number of nodes does not exceed a given bound. This can be used to keep the individuals relatively small in order to save run time.

4 Experimental Results

Experiments have been carried out to show the effectiveness of the approach. The runs presented in the following are supposed to exemplify the different working methods of the evolutionary algorithm. The first one starts with a population of randomly initialized graphs, the second one starts with correct OBDDs.

Example 4. Consider the *hidden weighted bit* (HWB) function given by the following equation:

$$HWB(x_0,\dots,x_{n-1}) = \begin{cases} 0 & \text{if } w = 0 \\ x_{w-1} & \text{else} \end{cases} \quad \text{where} \quad w = |x_0,\dots,x_{n-1}|.$$

HWB is known to have only OBDD representations of exponential size in the number of variables, but FBDDs of quadratic size [13]. In this example the presented approach is used to evolve a BDD that represents the 4 bit HWB function. The OBDD is shown in Figure 9 on the left. The parameters are set up as follows: a deterministic tournament selection of size 2 and a comma replacement combined with weak elitism are used. The population contains 100 individuals which are initialized randomly with an initial depth of 6. The following genetic operators are applied: `OutputXover` with a rate of 0.2, `VariableMut`, `EdgeSwapMut` and `EraseNodeMut` each with a rate of 0.05 and `CombRestucturing` with a rate of 0.2. The optimal BDD shown in Figure 9 on the right could be evolved after 90 generations.

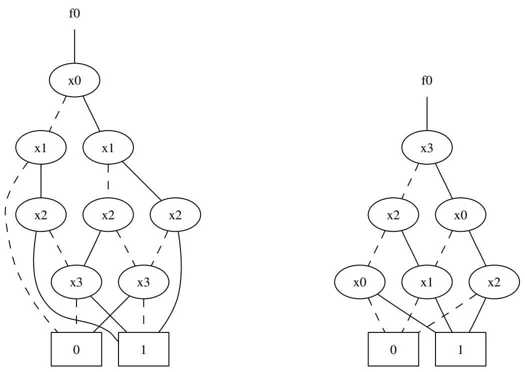


Fig. 9. OBDD and BDD for the 4 bit hidden weighted bit function

Example 5. In [2] Bryant introduced a function $f(x_0, \ldots, x_{2n-1}) = x_0 \cdot x_1 + x_2 \cdot x_3 + \cdots + x_{2n-2} + x_{2n-1}$ that is sensible to the variable ordering, i.e. the size of its OBDD varies between linear and exponential. In this example we consider a benchmark function $g : \mathbb{B}^6 \to \mathbb{B}^4$, where the four outputs are variants of Bryant's function:

$$\begin{aligned} g_0(x_0, \ldots, x_5) &= x_0 \cdot x_1 + x_2 \cdot x_3 + x_4 \cdot x_5 \\ g_1(x_0, \ldots, x_5) &= x_0 \cdot x_2 + x_1 \cdot x_3 + x_4 \cdot x_5 \\ g_2(x_0, \ldots, x_5) &= x_0 \cdot x_3 + x_1 \cdot x_4 + x_2 \cdot x_5 \\ g_3(x_0, \ldots, x_5) &= x_0 \cdot x_5 + x_1 \cdot x_4 + x_2 \cdot x_3 \end{aligned}$$

Note that for each output function there is a different order leading to the optimal OBDD size, thus there is no global order for which all partial OBDDs are optimal. The size of the optimal shared OBDD is 31. The evolutionary approach is applied with the following settings: the initial population consists of 100 correct OBDDs. `OutputXover` and `CombRestucturing` are applied each with a rate of 0.3, `TautoReduction` and `SplitReduction` are applied with a respective rate of 0.1. Selection and replacement are the same as in Example 4. After 41 generations, an individual emerged that represents the target function with 17 nodes.

Table 2 shows additional results. Besides the name of the circuit and the numer of inputs and outputs the size of the minimal OBDD is given. The last column shows the size of the smallest BDD that could be found by our approach. The algorithm has been run 50 times with a limit of 200 generations and a population size of 100. Only for the largest benchmarks with 7 inputs a population size of

Table 2. Benchmark circuits

circuit	i/o	OBDD	GA	circuit	i/o	OBDD	GA
rnd_3_3_a	3/3	9	7	hwb4	4/1	7	6
rnd_3_3_b	3/3	7	7	hwb5	5/1	14	12
rnd_3_3_c	3/3	8	7	hwb6	6/1	21	20
rnd_3_3_d	3/3	9	8	isa2	5/1	10	8
rnd_4_2_a	4/2	10	11	isa3	10/1	26	20
rnd_4_2_b	4/2	12	11	f51m.49	3/1	4	4
rnd_4_2_c	4/2	11	12	cm82a.f	3/1	5	5
rnd_4_2_d	4/2	12	11	b1	3/4	8	8
rnd_5_1_a	5/1	11	11	f51m.48	4/1	6	6
rnd_5_1_b	5/1	11	11	cm42a.e,f	4/2	5	5
rnd_5_1_c	5/1	12	11	cu.pq	4/2	10	10
rnd_5_1_d	5/1	12	12	cm82a.h	5/1	7	7
rnd_5_4_a	5/4	37	35	C17	5/2	7	7
rnd_5_4_b	5/4	35	34	cm138a.m	6/1	6	6
rnd_5_4_c	5/4	34	32	pcl3.3	7/1	8	8
rnd_5_4_d	5/4	39	38	cu.rs	7/2	8	8

200 and a generation limit of 300 have been used. As can be seen, in many cases smaller representations could be found. Especially the HWB and ISA functions for which there is an exponential gap between their OBDD- and BDD-size show good results. But also for randomly generated functions the graph size could be improved. For other benchmarks no improvements could be made, but it should be noted that for numerous common functions there are OBDDs of linear size and thus no improvements can be expected by using BDDs instead.

5 Conclusions and Future Work

In this paper it has been shown that it is possible to construct compact BDDs using genetic programming. First experiments have yielded some promising results. However, there are still some problems to be solved. For large functions that depend on many variables, it takes too long to evolve a correct solution from a randomly initialized population. This can be avoided, if correct OBDDs are used as initial population. Unfortunately, the regular structure of the OBDDs seems to be very stable, and the algorithm will hardly escape from the local optima induced by the OBDDs. Certainly there is still capability for improvements. Possibly new operators that act less locally than `Restructuring` could help to "break" the OBDDs.

References

1. Drechsler, R., Günther, W.: Towards One-Path Synthesis. Kluwer Academic Publishers (2002)
2. Bryant, R.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Comp. **35** (1986) 677–691
3. Ajtai, M., Babai, L., Hajnal, P., Komlos, J., Pudlak, P., Rödl, V., Szemeredi, E., Turan, G.: Two lower bounds for branching programs. In: Symp. on Theory of Computing. (1986) 30–38
4. Bryant, R.: On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication. IEEE Trans. on Comp. **40** (1991) 205–213
5. Becker, B., Drechsler, R., Werchner, R.: On the relation between BDDs and FDDs. Technical Report 12/93, Universität Frankfurt, 12/93, Fachbereich Informatik (1993)
6. Kebschull, U., Schubert, E., Rosenstiel, W.: Multilevel logic synthesis based on functional decision diagrams. In: European Conf. on Design Automation. (1992) 43–47
7. Drechsler, R., Sarabi, A., Theobald, M., Becker, B., Perkowski, M.: Efficient representation and manipulation of switching functions based on ordered Kronecker functional decision diagrams. Technical Report 14/93, J.W.Goethe-University, Frankfurt (1993)
8. Ashar, P., Ghosh, A., Devadas, S., Newton, A.: Combinational and sequential logic verification using general binary decision diagrams. In: Int'l Workshop on Logic Synth. (1991)

9. R.Rudell: Dynamic variable ordering for ordered binary decision diagrams. In: Int'l Workshop on Logic Synth. (1993) 3a–1–3a–12
10. Drechsler, R., Becker, B., Göckel, N.: A genetic algorithm for variable ordering of OBDDs. IEE Proceedings **143** (1996) 364–368
11. Sakanashi, H., Higuchi, T., Iba, H., Kakazu, Y.: Evolution of binary decision diagrams for digital circuit design using genetic programming. In: ICES. (1996) 470–481
12. Bollig, B., Wegener, I.: Improving the variable ordering of OBDDs is NP-complete. IEEE Trans. on Comp. **45** (1996) 993–1002
13. Wegener, I.: Bdds – design, analysis, complexity, and applications. Discrete Applied Mathematics **138** (2004) 229–251
14. Friedman, S., Supowit, K.: Finding the optimal variable ordering for binary decision diagrams. In: Design Automation Conf. (1987) 348–356
15. Buch, P., Narayan, A., Newton, A., Sangiovanni-Vincentelli, A.: On synthesizing pass transistor networks. In: Int'l Workshop on Logic Synth. (1997)
16. Ferrandi, F., Macii, A., Macii, E., Poncino, M., Scarsi, R., Somenzi, F.: Layout-oriented synthesis of PTL circuits based on BDDs. In: Int'l Workshop on Logic Synth. (1998) 514–519
17. Koza, J.: Genetic Programming - On the Programming of Computers by means of Natural Selection. MIT Press (1992)
18. M. Keijzer, J.J. Merelo, G.R., Schoenauer, M.: Evolving objects: a general purpose evolutionary computation library. In: Evolution Artificielle. (2001)
19. Somenzi, F.: CUDD: CU Decision Diagram Package Release 2.4.0. University of Colorado at Boulder (2004)

Efficient Evolutionary Approaches for the Data Ordering Problem with Inversion

Doina Logofatu and Rolf Drechsler

Institute of Computer Science, University of Bremen,
Bremen 28359, Germany
{doina, drechsle}@informatik.uni-bremen.de

Abstract. An important aim of circuit design is the reduction of the power dissipation. Power consumption of digital circuits is closely related to switching activity. Due to the increase in the usage of battery driven devices (e.g. PDAs, laptops), the low power aspect became one of the main issues in circuit design in recent years. In this context, the Data Ordering Problem with and without Inversion is very important. Data words have to be ordered and (eventually) negated in order to minimize the total number of bit transitions. These problems have several applications, like instruction scheduling, compiler optimization, sequencing of test patterns, or cache write-back. This paper describes two evolutionary algorithms for the Data Ordering Problem with Inversion (DOPI). The first one sensibly improves the Greedy Min solution (the best known related polynomial heuristic) by a small amount of time, by successively applying mutation operators. The second one is a hybrid genetic algorithm, where a part of the population is initialized using greedy techniques. Greedy Min and Lower Bound algorithms are used for verifying the performance of the presented Evolutionary Algorithms (EAs) on a large set of experiments. A comparison of our results to previous approaches proves the efficiency of our second approach. It is able to cope with data sets which are much larger than those handled by the best known EAs. This improvement comes from the synchronized strategy of applying the genetic operators (algorithm design) as well as from the compact representation of the data (algorithm implementation).

Keywords: Evolutionary Algorithms, Digital Circuit Design, Low Power, Data Ordering Problem, Transition Minimization, Optimization, Graph Theory, Complexity.

1 Introduction

The power consumption became a barrier during the design of embedded systems, as soon as the limits of the paradigm “the smaller the device, the faster it is” were reached. Due to the ever increasing demand for electronic devices with bigger storage capacity and quicker access time (see e.g. [5]), the low-power techniques have to be taken into account already in the first phases of the design process. Thus, various methods for decreasing the power dissipation have been developed (see e.g. [1], [15], [21], [26]).

The challenges in the design of embedded systems can be split into two major categories: hardware-related and software-related. The hardware designers try to find methods to optimize the switching activity and the voltage in the circuit ([6], [25]). The software component is also very important for the power consumption of the circuit ([23], [24]). In this area an efficient design can provide significant improvements. Power consumption often is directly determined by the design complexity. For this reason, power consumption has grown in the past years with increasing design complexities. Therefore, power management is a critical design priority. As such, lower power consumption has a positive effect on battery life, packaging, cooling costs and reliability. A new direction of the design methodologies is necessary to handle the power management issue in a successful way.

The power consumption on the software level depends on the switching activity and the capacitance. The switching activity, as an important design metric, characterizes the quality of an embedded system-on-chip design. It is implicitly related to the orderng of the data sequences. The first problem engaged with this topic is the ordering of data words to minimize the total number of transitions *Data Ordering Problem* (DOP). In [18] it is demonstrated that this problem is NP-complete. Recently, some algorithms were proposed for optimizing the number of transitions. In [22], Stan and Burleson introduced the *bus-invert method*. The main idea is to use an extra bus line with bits, called *invert*, which contains the information regarded as the *phase-assignment* for all transmitted words. For every word there is a bit flag that signals if the transmitted word is complemented (inverted), flag = 1, or left as initial, flag = 0. Adding this new paradigm which applies to DOP an extra degree of freedom, the total number of transitions can be lower than the number provided with DOP. The resulting problem is the so-called *Data Ordering Problem with Inversion* (DOPI) (see also [11, 19]). A formal definition, related terms, and examples are given in Section 2.

As a general method for solving optimization problems, EAs are getting more and more popular. Recently, EAs have been successfully applied to several problems in VLSI CAD (see e.g. [8], [9], [11], [16]). In [10] an efficient genetic approach for the DOP is proposed and in [11] an evolutionary algorithm for DOPI. This EA approach provides high-quality results (better than polynomial heuristics), but they need also a large amount of runtime.

In this paper, we propose two evolutionary algorithms for DOPI: one which performs quick small improvements and the other one, which is a hybrid genetic algorithm that performs significant improvements in a larger amount of run time. For smaller-sized instances we applied the optimal exact algorithms (both DOP and DOPI) for comparing the behavior of the two problems as well as the results provided by other related algorithms. In [19] a lower bound algorithm for DOPI is introduced. In our study, we will use it to check the deviation from the optimum of the results provided with the proposed EAs. There are three categories of input data: small, medium and large data sizes. The focus is on optimizing the DOPI approach, which adds the paradigm of phase-assignment to DOP and thereby improves the performance of the results.

2 DOP and DOPI Problems Domain

Definition 2.1. *Hamming distance.* If a word w_r is transmitted immediately followed by w_s, the total number of transitions is given by the number of bits that change. This is

$$d(w_r, w_s) = \sum_{j=1}^{k} w_{rj} \oplus w_{sj} ,$$

also known as the *Hamming distance* between w_r and w_s. Here, the w_{rj} denotes the j^{th} bit of w_r, and $\oplus$ the XOR operation. For instance, *d(1010, 0100) = 3*. Word reordering can change the number of transitions significantly.

Definition 2.2. *Total number of transitions.* The *total number of transitions* is the sum of the Hamming distance needed for the transmission of all the words. It is denoted with N_T. If σ is a permutation of the bit strings $w_1, w_2, ..., w_n$, than the total number of transitions will be: $N_T = \sum_{j=1}^{n-1} d(w_{\sigma(j)}, w_{\sigma(j+1)})$.

Definition 2.3. *Adjacency matrix.* For a given problem instance (where n is the number of words and k their length) we define the *adjacency matrix* $A_{n\,x\,n}$ where $A(i, j) = d(w_i, w_j)$.

Definition 2.4. *Phase-assignment.* The inversion of a data word w_i is also called the negation (complementation) and is denoted $\overline{w_i}$. The *polarity* (phase-assignment) δ is a function defined on the set of words with values in a set of words and $\delta(w)$ is $\overline{w}$ (case w may be complemented) or w (case w may be not complemented).

Proposition 1.1. For adjacency matrix and inversion holds $\forall i, j \in \{1, .., n\}$:

a. $d(w_i, w_j) = d(w_j, w_i) \Rightarrow a_{ij} = a_{ji}$

b. $d(w_i, w_i) = 0 = \ a_{ii} = 0$

c. $d(w_i, w_j) = d(\overline{w_i}, \overline{w_j})$

d. $d(w_i, \overline{w_j}) = d(\overline{w_i}, w_j) = k - d(w_i, w_j) = k - a_{ij}$

Formalized, the definitions of the DOP and DOPI will be:

Definition 2.5. *DOP*: Find a permutation σ of the bit strings $w_1, w_2, ..., w_n$ such that the total number of transitions:

$$N_T = \sum_{j=1}^{n-1} d(w_{\sigma(j)}, w_{\sigma(j+1)}) \quad (1)$$

is minimized.

Definition 2.6. *DOPI*: Find a permutation σ of the bit strings $w_1, w_2, \ldots, w_n$ and a phase-assignment δ such that the total number of transitions:

$$N_T = \sum_{j=1}^{n-1} d(\delta(w_{\sigma(j)}), \delta(w_{\sigma(j+1)})) \tag{2}$$

is minimized.

3 Previous Approaches

The DOP and DOPI are very similar to the *Traveling Salesman Problem* (TSP). For all three problems a good ordering of elements with respect to a given weight between each two elements has to be determined. Since the DOP and DOPI are NP-Complete, the exact algorithms can only handle very small instances. In the past few years some heuristics were developed for both DOP and DOPI (most of them in relation with the TSP problem):

1. *Double Spanning Tree* (DST) [12]
2. *Spanning Tree/Minimum Matching* (ST-MM) [12]
3. *Greedy Min* (GM) [18]
4. *Greedy Simple* (GS) [10]
5. *Evolutionary Heuristics* [11]

The most powerful polynomial heuristic known so far is *Greedy Min* and it can be applied to both DOP/DOPI:

1) Computes the Hamming distance for all (distinct) pairs of given words and selects the pair with a minimum cost.

2) Chooses the most convenient pair of words. The beginning sequence will contain these two words.

3) Builds progressively the sequence, adding in every step of the most convenient word (that was not yet added). This word can be added either at the beginning or at the end of the sequence, depending where the Hamming distance is minimal.

The EAs are the best algorithms regarding the quality of results. Such evolutionary approaches provide better results than the above-presented *Greedy Min*, but with significantly more time resources. EAs which perform high-quality optimizations are presented in [10], [11]. In [11] are presented evolutionary algorithms for both DOP and DOPI. For DOPI the mutation and crossover operators are applied in parallel for creating new individuals. This is also a hybrid EA, since the initial individuals are preprocessed using greedy methods. The results provided by the EA are better than the *Greedy Min* results, but the maximal number of words is 100.

In [19] a graph theory related model for DOPI is introduced, together with a relevant graph theoretic background. For a DOPI instance with n words, each of length k, a multigraph can be created accordingly. The vertices are the words and the edges are labeled with the distance between the words. According to *Proposition 1.1. c)* and *d)*, if the words are in the same phase-assignment (both *0* or both *1*), then the distance between them is the same. Also, if they are in different phase-assignment, the distance remains the same. There are two edges between two vertices (one if the words are in the same phase, one for the case if they are transmitted in different phases). In this manner DOPI is transformed in the equivalent NP-complete problem of finding the Hamiltonian path with the minimum length. As shown in [19], a *lower bound* for the length of this path is the weight of the minimal spanning tree of the multigraph. To determine the minimal spanning tree there are two classical greedy algorithms: *Prim* (uses the vertex connections) and *Kruskal* (uses the edges). In the experimental tests from Section 5, we use the Kruskal approach to check the deviation to the optimum for the provided EA results.

4 Evolutionary Algorithms

In this section the two evolutionary approaches for DOPI will be presented: the first is a simple one which operates on a single individual (the *Greedy Min* resulted one) with the *Simple Inversion Mutation* (SIM) operator and the *Simple Cycle Inversion Mutation* (SCIM) operator. This performs improvements of the greedy solution in a small amount of time. The second one is based on the classical genetic algorithm model. The genetic operators are applied synchronized for ordering and phase-assignment, in order to preserve the good subsequences. As can be seen later in the result tables, the results are much better than the previous mutation algorithm, but the time needed increased significantly. follow these instructions closely in order to make the volume look as uniform as possible.

We would like to stress that the class/style files and the template should not be manipulated and that the guidelines regarding font sizes and format should be adhered to. This is to ensure that the end product is as homogeneous as possible.

4.1 Overview of Genetic Algorithms

A Genetic Algorithm (GA) is an optimization method with simple operations based on the natural selection model [17]. Genetic algorithms have been applied to hard optimization problems including VLSI layout optimization, boolean satisfiability, and the Hamiltonian circuit problem ([13], [16], [20]). There are four main distinctions between GA-based approaches and traditional problem-solving methods:

a) GAs operate with a genetic representation of potential solutions, not the solutions themselves.
b) GAs search for optima of a population of potential solutions and not a single solution (the genetic operators alter the composition of children).
c) GAs use evaluation functions (*fitness*), no other auxiliary knowledge such as derivative information used in the conventional methods.
d) GAs use probabilistic transition rules (not deterministic rules) and various parameters (population size, probabilities of applying the genetic operators, etc.)

For a specific problem, it is very important to use related genetic operators, which preserve the good traits from the parents, but are also able to bring improvements in the resulting children. The initialization step and the parameter settings are also very significant.

4.2 A Simple Mutation Algorithm for DOPI

The EV_MUT_ALGORITHM (MUT) is an evolutionary algorithm, which operates successively on an individual. The *Greedy Min* algorithm generates the beginning individual. The pair *(Permutation, BitString)* is a potential solution. The *Permutation* denotes the *ordering* and *BitString* denotes the *phase-assignment*. The mutation will be applied for two random cut points to the both components *(Permutation* and *Bit-String)*. The new individual will substitute the current best, if it is better. The used mutation operators are the classical *SIM* and a derivative thereof, denoted with *Simple Cycle Inversion Operator* (SCIM).

4.2.1 Simple Inversion Mutation (SIM)

SIM was introduced in 1975 by Holland [14]: 2 cut points are randomly selected and the subsequence between them is mirrored. For example, for the permutation *(1, 2, 3, 4, 5, 6, 7, 8)* and the cut points *3* and *5* the result will be the permutation *(1, 2, 5, 4, 3, 6, 7, 8)*. The same will be applied also to the bit string, with the same cut points.

This operator can improve the current solution, which was constructed on a local optimum (greedy) basis. In the DOPI case the same cut points are considered, as well as for the permutation. Using this operator the sequences inside the cut points for the current individual are preserved. As a consequence, the improvement will come from the sum of the distances in the two interior cut points regions, which can be reduced.

4.2.2 Simple Cycle Inversion Mutation (SCIM)

This operator additionally includes the possibility to change the extremities of the potential solution. A potential solution is seen as a circular structure, in which the last element, had it been in a linear structure, it would not have had a successor. As such, by turning the linear structure into a circular one, the last element is tied to the first element, the latter becoming the successor of the former. This way, the last element obtains a successor - namely the first element of the former linear structure.

This potential solution is a circular structure, where the next for the last element is the first one. Then, if the cut points are *(i, j)* with $i > j$, we will invert the sequence from the places $i, i+1, ..., n, 1, ..., j$. The elements from the positions $j+1, j+2,..., i-1$ will remain untouched.

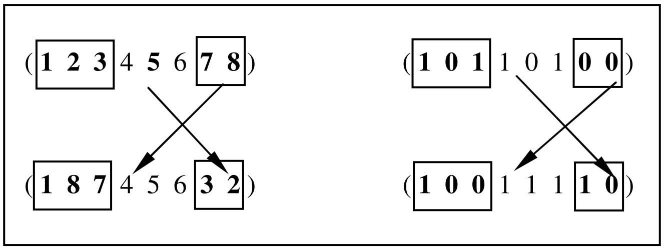


Fig. 1. The mutation operator SCIM for a Permutation and Bit String and the cut points (7, 3)

Note: in case of the permutation, the sequence *(7, 8, 1, 2, 3)* is transformed in the sequence *(3, 2, 1, 8, 7)* and the places are preserved. The same happens in the case of the bit string. The use of the operator yields improvements to the permutation due to the transformation in the cut points as well as to the change of the end points. Because there are *n(n-1)/2* different pairs of words, the total number of performed operations is also *n(n-1)/2*. The cut points are chosen randomly. If the total number of performed operations grows, the provided result is expected to be better.

4.3 A Genetic Algorithm for DOPI

The GENETIC_ALGORITHM (GA) is based on the classical sketch of a genetic algorithm, in which a part of the initial population is initialized using the two greedy methods and another part is formed from random individuals. On the current population, mutation and crossover operators are applied with a given probability, and the best *populationSize* (the population individuals' count) individuals are kept in the selection phase.

4.3.1 Representation

A potential solution is a pair (Permutation, BitString). The Permutation object is the representation of the word ordering. The BitString object is the representation of the phase-assignment (value 1/0 means that the corresponding data word has/not to be inverted). A population is a set of such pair-elements. The genetic operators can be applied on these elements.

The adjacency matrix is symmetric and its main diagonal is always filled with *0s* (zeros). We will only keep its elements from the upper main region in a vector with $(n-1) + (n-2) + \ldots + 1 = n(n-1)/2$ elements (instead of the whole matrix with n^2 elements). This representation will minimize the space needed for its representation and assures a higher speed for the application.

4.3.2 Objective Function and Selection

The value given in formula (2) is used as an objective function that measures the fitness of an element.

The selection is performed by *roulette wheel selection*. The best *populationSize* elements are chosen for the next generation. This strategy guarantees that the best element never gets lost and a fast convergence is obtained.

4.3.3 Initialization

Often it is helpful to combine EAs with problem-specific heuristics (see e.g. [4], [8], [9], [11]). The resulting EAs are called Hybrid EAs. In our case, we will use Greedy specific techniques for initialization. Some of the initial individuals will be initialized using the Greedy Min method, some of them with a different Greedy technique - with comparable results. The rest will be random individuals. The initialization using the greedy methods guarantees that the starting point is not completely random and thereby the convergence is speeded up. Because the complexity of the Greedy algorithms is polynomial O(n2), this initialization step uses little time, but helps significantly for a faster convergence. The number of initial Greedy Min generated individuals is n, the same as the number of individuals generated with the second Greedy method (like Greedy Min, with the difference that it is restricted on one end).

4.3.4 Genetic Operators

The used crossover operators are the classical PMX [13] and OX [3] and two derived ones: *Cycle PMX* (CPMX) and *Cycle OX* (COX).

PMX: Constructs the children by choosing the part between the cut positions from one parent and preserve the absolute position and order of as many variables as possible from the second parent. The PMX can be corresponding applied to a bit string, preserving the inner sequence from a parent and the lateral ones from the other one.
CPMX: Constructs the children exactly like in PMX with the difference that the next element for the last one is the first one, like in the SCIM. The PMX is applied to the sequence from the places *(i, j)* with *i>j*: *i, i+1, ..., n, 1, ...j.* The CPMX can be also applied to a bit string with the preservation of this sequence from a parent and the inner one from the other parent.
OX: Construct the children by choosing the part between the cut positions from one parent and preserve the relative position and order of as many variables as possible from the second parent.
COX: It is similar with SCIM and CPMX, the sequence is considered circular and the OX will be applied to the pairs *(i, j)* with *i>j*, for the elements from places *i, i+1, ..., n, 1, ...j.* It can be applied also to bit strings.

The used mutation operators are the SIM and SCIM, which are presented in the above section.

4.3.5 Algorithm

The algorithm is based on the classical sketch of the genetic algorithms. A very important step is the initialization of the first individuals. A refined version of the classical genetic algorithm:

```
ALGORITHM_GA_DOPI
        Initialize(populationSize)
        Initialize(crossoverRate)
        Initialize(mutationRate)
        numCrossovers ← populationSize*crossoverRate
        numMutations ← populationSize*mutationRate
        Initialise_GreedyMin_individuals()
        Initialise_Greedy1_individuals()
        Initialise_Random_individuals()
        for( i ← 1;i ≤numGenerations; step 1) execute
                Apply_Crossover_operators(numCrossovers);
                Apply_Mutation_operators(numMutatios);
                Calculate_fitness(allNewIndividuals);
                Remove_WorstInviduals (numCrossovers+numMutations);
        end_for
        return best_element;
END_ALGORITHM_GA_DOPI
```

Fig. 2. Sketch for the DOPI hybrid EA

For the *Greedy Min* initialization step *n-1* different pairs of words are chosen and a solution is built using the local optima strategy given by *Greedy Min*. The same will be performed for another *Greedy* individuals' initialization, with the difference that at the beginning there are only different words generated, which are supposed to remain the start words for the potential solutions.

4.3.6 Parameters Settings

The chosen settings are based on experimental tests. Since the genetic algorithm is applied to the different data sizes, from very small to large ones, it becomes necessary to adapt these settings to the size of the problem. In [11] the population size is constantly 50, since the maximal treated problem size is 100. The stop condition of the algorithm [11] is the moment when no improvement occurs for 5000 generations. In our case, the necessary time to create and process a new generation for large data sets is very high. That's why the number of generations is also related to the input data size.

For the small data examples ($n = 6, 7, 8, 9, 10$) the size of the population is set to *100*, the number of generations is *150*. For the medium examples ($n = 50, 100, 150$) the population size is $5n$ and the number of generations is $20n$. For the big examples ($n = 400, 600, 800$) the population size is $2n+n/2$ and the number of generations is $3n$. The mutation rate is *0.2*; that means that the total number of new individuals obtained with mutation is *populationSize*0.2*. The fertility rate is *0.5*; it follows that at every generation the number of individuals obtained with crossover operators are *populationSize*0.5*. This fertility rate is higher because experiments have shown that the crossover operations perform improvements more often than mutation operations. The crossover and mutation operators are uniformly considered.

5 Experimental Results

All algorithms are implemented in C++, using the *Standard Template Library*. The experimental results are focused on the DOPI problem, which is more efficient than DOP (due to the extra degree of freedom in choosing the phase-assignment). The extra degree of freedom increases also the complexity of the problem. The DOP complexity is $O(n!)$, since for a given n there are $n!$ different permutations. The complexity for DOPI is $O(2^n n!)$, because for every permutation all the bit strings of length n have to be checked (there are 2^n such bit strings). It follows that the exact solutions can be found only for very small dimensions. The test program initializes a random instance for the problem with the given (n, k) values: n bit strings (the words), each of length k. For these instances, various algorithms are applied. In the first table we perform the exact algorithms. For comparing the DOP and DOPI representative results, this table contains also the DOP random and exact ones. The sizes are $n \in \{6, .., 10\}$ and for every such n value, $k \in \{10, 30, 50\}$. The performed algorithms are: *RAN* (random), *EX* (exact), *G* (Greedy Min), *MUT* (EV_MUT_ALGORITHM), *GA* (Genetic Algorithm) and *LB* (Lower Bound).

Table 1. Small examples

Input Data		DOP		DOPI					
n	k	RAN	EX	RAN	EX	G	MUT	GA	LB
6	10	25	18	24	14	15	14	14	13
6	30	71	61	72	58	59	59	58	58
6	50	120	103	135	97	99	98	97	96
7	10	30	17	37	16	18	17	16	16
7	30	85	72	90	67	71	70	68	66
7	50	146	129	149	122	124	123	122	120
8	10	35	19	39	13	15	14	13	12
8	30	100	85	113	78	80	78	78	75
8	50	176	150	172	145	148	147	146	143
9	10	33	25	36	25	27	27	26	24
9	30	125	93	121	86	90	88	87	84
9	50	187	159	185	149	152	150	150	146
10	10	44	36	41	24	29	26	25	22
10	30	142	109	127	99	102	100	101	95
10	50	236	197	238	168	173	170	169	165

Table 2. Medium examples

Input Data		DOPI				
n	k	RAN	G	MUT	GA	LB
50	115	2856	2300	2292	2267	2214
50	215	5355	4449	4438	4396	4354
50	315	7766	6787	6773	6728	6655
100	115	5633	4423	4408	4361	4278
100	215	10696	8820	8811	8765	8678
100	315	15604	13479	13470	13398	13286
150	115	8425	6537	6530	6487	6327
150	215	15976	13237	13221	13161	12905
150	315	23425	20176	20158	20095	19884

Table 3. Large examples

Input Data		DOPI				
n	k	RAN	G	MUT	GA	LB
400	200	40008	31988	31986	31925	31338
400	600	119862	106269	106265	106171	105066
400	1000	199609	182551	182534	182371	181147
600	200	59911	47635	47629	47495	46719
600	600	180141	158660	158658	158507	157016
600	1000	299344	273182	273168	272993	271145
800	200	79973	62948	62940	62795	61757
800	600	239407	210774	210772	210689	208647
800	1000	399767	363257	363248	363119	360553

As expected, all exact values for DOPI are better than the values for exact DOP and the same instances. The *lower_bound* values for DOPI are lower than the exact values, but very close to them. The mutation algorithm, which is not time consuming and changes progressively the current individual, often produces improvements. The genetic algorithm is in general better than the mutation algorithm and often provides the best solution, especially for very small sizes.

We note that the results for GA are better than the mutation algorithm results. For the same *n*, if the *k* value is greater, then the difference from this result and the lower bound result is greater.

For these large examples the improvements provided with the MUT algorithm are not as significant as with the previous data sets. The genetic algorithm still provides the best solutions and the difference to the *lower_bound* algorithm remains small related to the *(n, k)* instance sizes. Because the complexity of the problem is exponential, it is expected that if the size of the population and the number of generations are greater, the results for the large examples will be even better. This requires a powerful system for running the GA application.

6 Conclusions

Two evolutionary approaches for DOPI were presented. Additionally, we described algorithms such as: *random*, *exact*, *lower_bound* and *greedy* which helped to compare the results of these approaches. Due to the very high complexity of the problem, it is very important that we know how the genetic operators are applied. The pair *(Permutation, BitString)* to represent a DOPI solution is considered an object and the operators have to be applied accordingly to both components, in order to preserve the good traits from the parents as well as to have the chance to provide better successors. The genetic algorithm is a hybrid one, because the initialization step uses greedy techniques. In addition to the known *Greedy Min* algorithm, another comparable Greedy is introduced and used during the initialization phase. The complexity for larger data sets comes from two directions: one that is the needed space to store the whole population and one is the time to perform the genetic operations. The final results provided by both EAs are better than the results provided by the Greedy Algorithm.

References

1. Chandrakasan, A. P., Potkonjak, M., Rabaey, J., Brodersen, R. W.: HYPER-LP: a system for power minimization using architectural transformations. In Int'l Conf on CAD, pages 300-303, 1992
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.: Introduction to Algorithms, Second Edition, The MIT Press; 2nd edition (September 1, 2001)
3. Davis, L.: Applying adaptive algorithms to epistatic domains. In Proceedings of IJCAI, pages 162-164, 1985
4. Davis, L.: Handbook of Genetic Algorithms. van Nostrand Reinhold, New York, 1991
5. De Micheli, G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill, Inc., 1994
6. Devadas, S., Malik, S.: A survey of optimization techniques targeting low power VLSI circuits. In Design Automation Conf., pages 242-247, 1995

7. Drechsler, N., Drechsler, R.: Exploiting don't cares during data sequencing using genetic algorithms. In ASP Design Automation Conf., pages 303-306, 1999
8. Drechsler, R.: Evolutionary Algorithms for VLSI CAD. Kluwer Academis Publisher, 1998
9. Drechsler, R., Drechsler, N.: Evolutionary Algorithms for Embedded System Design. Kluwer Acadmeic Publisher, 2002
10. Drechsler, R., Göckel, N.: A genetic algorithm for data sequencing. Electronic Letters, 33(10): 843-845, 1997
11. Drechsler, R., Drechsler, N.: Minimization of Transitions by Complementation and Resequencing using Evolutionary Algorithms, In Proceedings of 21st IASTED International Multi-Conference Applied Informatics (AI 2003), IASTED International Conference on Artificial Intelligence and Applications (AIA 2003), Innsbruck, 2003
12. Garey, M. R., Johnson, D. S.: Computers and Intractability – A Guide to NP-Completeness. Freeman, San Francisco, 1979
13. Goldberg, D. E., Lingle, R.: Alleles, loci, and the traveling salesman problem. In Int'l Conference on Genetic Algorithms, pages 154-159, 1985
14. Holland, J. H.: Adaption in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI, 1975
15. Iman, S., Pedram, M.: Multilevel network optimization for low power. In Int'l Conf. On CAD, pages 372-377, 1994
16. Mazumder, P., Rudnick, E.: Genetic Algorithms for VLSI Design, Layout & Test Automation. Prentice Hall, 1998
17. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
18. Murgai, R., Fujita, M., Krishnan, S. C.: Data sequencing for minimum-transition transmission. In *IFIP Int'l Conf. on VLSI,* 1997
19. Murgai, R., Fujita, M., Oliveira, A.: Using complementation and resequencing to minimize transitions. In *Design Automation Conf.,* pages 694-697, 1998
20. Oliver, I. M., Smith, D. J., Holland, J. R. C.: A study of permutation crossover operators on the traveling salesman problem. In *Int'l Conference on Genetic Algorithms*, pages 224-230, 1987
21. Shen, W.-Z., Lin, J.-Y., Wang, F.-W.: Transistor reordering rules for power reduction in CMOS gates. In *ASP Design Automation Conf.*, pages 1-6, 1995
22. Stan, M., Burleson, W.: Limited-weight codes for low-power I/O. *Int'l Workshop on Low Power Design,* 1994
23. Tiwari, V., Malik, S., Wolfe, A., Lee, M.: Power analysis of embedded software: A first step towards software power minimization. In *Int'l Conf. on CAD*, pages 384-390, 1994
24. Tiwari, V., Malik, S., Wolfe, A., Lee, M.: Instruction level power analysis and optimization software. In VLSI Design Conf., 1996
25. Tsui, C., Pedram, M., Despain, A. M.: Technology decomposition and mapping targeting low power dissipation. In Design Automation Conf., pages 68-73, 1993
26. Vaishnav, H., Pedram, M.: PCUBE: A performance driven placement algorithm for low power design. In European Design Automation Conf., pages 72-77, 1993
27. Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesman: The genetic edge recombination operator. In Int'l Conference on Genetic Algorithms, pages 133-140, 1989

GRACE: Generative Robust Analog Circuit Exploration

Michael A. Terry, Jonathan Marcus, Matthew Farrell,
Varun Aggarwal, and Una-May O'Reilly

Computer Science and Artificial Intelligence Lab (CSAIL),
Massachusetts Institute of Technology,
Cambridge, MA, USA
m_terry@alum.mit.edu, unamay@csail.mit.edu

Abstract. We motivate and describe an analog evolvable hardware design platform named GRACE (i.e. Generative Robust Analog Circuit Exploration). GRACE combines coarse-grained, topological circuit search with intrinsic testing on a Commercial Off-The-Shelf (COTS) field programmable device, the Anadigm AN221E04. It is suited for adaptive, fault tolerant system design as well as CAD flow applications.

1 Introduction

With our Generative Robust Analog Ciruit Exploration (GRACE) tool we are investigating whether it is possible to evolve circuits that can be realized efficiently and in a routine manner. We are focusing upon the domain of analog circuit design. Our decision is motivated by the lack of automation in analog design as compared to digital design. We intend to investigate whether evolvable hardware (EHW) approaches can contribute to the complex, human-intensive activity domain of analog design.

The goal of this paper is to describe how we arrived at GRACE. By combining the exploitation of coarse grained elements with intrinsic testing, we think GRACE sits in an interesting and novel space. It allows a distinctive foray into on-line adaptive and fault tolerant evolvable hardware circuits since it uses a COTS (Commercial-Off-The-Shelf) device and standard components. This should make its results more acceptable to industry. It also allows an economical and time efficient foray into the broad domain of VLSI and CAD with its use of elements that are conversant with human design. We proceed thus: In Section 2 we describe how we reached a decision to select the Anadigm AN221E04 as GRACE's reconfigurable device. In Section 3 we give an overview of GRACE. In Section 4 we describe GRACE's genetic representation of a circuit and its search algorithms. In Section 5 we design a controller for a plant using GRACE to demonstrate its ability to evolve circuits. We conclude with a summary.

2 Choosing GRACE's Reconfigurable Device

For GRACE, the choice of its reconfigurable device was driven by three criteria (see [1] for a related discussion):

1. A desire to work at an abstraction level where the human design principles are *inherent* in the building blocks so that GRACE will derive conventional, human-competent, portable and robust circuits;
2. Availability of a reconfigurable device that matched the project's budget of $5K;
3. Flexibility that would allow design elements to be chosen depending upon the design problem, (i.e. level of hierarchy in the analog design flow).

Among the devices we assessed for our purposes were the class of custom designed Field Programmable Transistor Arrays (FPTA), the Lattice Semiconductor ispPAC10 field programmable analog array (FPAA, e.g. [2]), and the Anadigm FPAA family ([3]).

With respect to Criterion 1, there are open questions regarding the suitability of an FPTA for evolving conventional, human-competent circuits:

1. Can an FPTA be configured to respect certain design principles so that interconnections of the transistor-switched cells and inter-cell topology will constitute circuits that a human engineer will trust? Some of these design principles such as no floating gates could be encoded in the circuit construction and circuit modification functions of an evolutionary algorithm. However, not all expectations/insights (such as parasitic insensitive connections) can be mapped into rules.
2. Can the non-idealities arising from the switching elements in the FPTAs be circumvented to avoid reliability, portability and engineering acceptance issues? The FPTAs require electronic switches that are implemented by transmission gates. This adds parasitic non-linear resistance and capacitance, which results in delay, de-amplification and alteration in frequency domain properties. While some of the evolved circuits to date *use* these non-idealities of switches as an integral feature of the design [4], realistically this makes the evolved design idiosyncratic (i.e. unportable) and unreliable since the behavior of its switches is neither characterizable or replicable.
3. Is there sufficient flexibility for sizing? Industry practice is to explore sizing options as a means of balancing functional goals and specifications. Because it only has one size of transistor, the FPTA-2 ([1])is constrained in this respect.

FPTAs are well suited for the exploration of non-conventional realms of circuit design such as polymorphic circuits, extreme temperature electronics and fault tolerant circuits [5, 6, 7]. However, they are not well suited to our desire to explore robust, novel topologies of interpretable and portable circuits.

Circuit synthesis with opamps has straightforward and methodical design rules (which can be easily incorporated in the evolutionary algorithm) to ensure

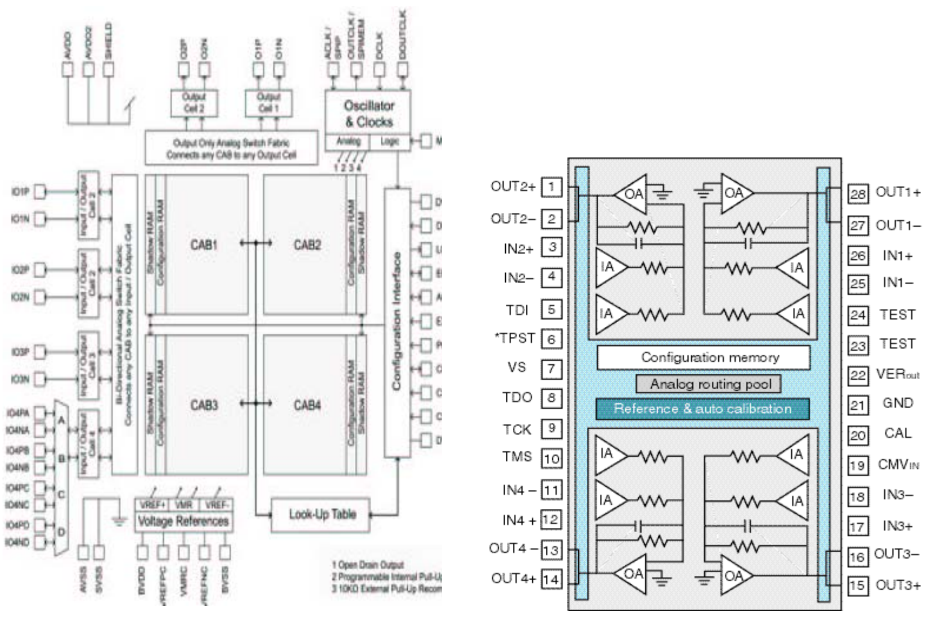


Fig. 1. Reconfigurable FPAA Architectures: Anadigm(Left), Lattice ispPAC10(right)

that the evolved circuit is interpretable and robust. The IsPAC10 and Anadigm FPAA have circuit elements based on opamps. The IsPAC10, see Figure 1 right, consists of 4 programmable analog modules (4 opamps, and 8 input amplifiers total) interconnected with programmable switching networks. Configuration of the IsPAC10 is a proprietary process [2] . The Anadigm AN221E04, see Figure 1 left (and described in more detail in Section 2.1), also provides opamp based circuits as building blocks. It uses switched-capacitor technology which is inherently robust and portable.

With respect to Criterion 2, the cost of an FPTA is beyond $5K. The development board of an IsPAC10 or Anadigm AN221E04 has a cost in the low hundreds of dollars. Integrated with a conventional computer and other signal processing devices, they facilitate a system with cost below $5K.

Wth respect to Criterion 3, each device we assessed offers a different level of circuit element granularity. The FPTAs are very flexible, fine-grained devices The U. of Heidelberg's FPTA ([8], henceforth called FPTA-H) is a switched network of 256 (16 X 16) programmable CMOS transistors (half NMOS and half PMOS) arranged in a checkerboard pattern.

The FPTA-1 designed at JPL ([4, 9]) is composed of 12 cells, where each cell contains 8 CMOS transistors interconnectable via 24 switches. The transistors are fixed size and the switches are electronically programmable. The FPTA-1 appears to have been a prototype device for FPTA-2. The FPTA-2 ([1]) contains an 8X8 matrix of 64 reconfigurable cells, where each cell consists of 14 transistors interconnectable via 44 switches. The transistors are fixed size. Each cell also contains three capacitors, two reconfigurable resistors and photodetectors. It fits into the Evolution-Oriented Reconfigurable Architecture (EORA) and is integrated with a DSP processor running the evolutionary algorithm to form the SABLES (Stand-Alone Board-Level Evolvable System).

FPTA-H is more versatile than FPTA-2 with respect to interconnection and sizing of transistors. In FPTA-H, in general any transistor can connect to any other transistor, while in FPTA-2, transistors are arranged in a particular topology with switches to realize different circuits. Even though the FPTA-2 cell has 44 switches which creates a large space of possible realizable topologies, there are human-conceivable designs which cannot be directly synthesized using it. In FPTA-H, 75 different aspect ratios could be chosen for each transistor, while the FPTA-2 uses fixed length transistors. This flexibility of FPTA-H comes at the cost of space (equivalent to the number of transistors that can be fabricated on the same chip). The FPTA-2 has 3.5 times more transistors on the same chip as compared to FPTA-H. This difference may also be attributed to the fact that FPTA-2 uses 0.18μm process, while FPTA-H uses 0.6μm process.

The IsPAC10 and Anadigm AN221E04 exemplify a tradeoff between flexibility and appropriate building block abstraction. The opamp is a building block that can be combined with passive components to arrive at variety of human-designed circuits such as amplifiers, integrators, differentiators, sum-difference amplifiers, or filters. However, it is not as flexible as a switched transistor array. On the IsPAC10, there is very limited interconnect between a small quantity of resources. The Anadigm AN221E04 provides a fixed abstraction level of opamp based circuits but supports very flexible interconnection.

After our assessment, we chose the Anadigm AN221E04 over the IsPAC10 or an FPTA. In a nutshell, we have forgone a large degree of flexibility by choosing a fixed abstraction level (of opamp based circuits) in order to ensure robustness, portability and reliability. Nonetheless we are content given that there are a number of analog design problems (such as PID controllers, ADCs and filters) which can be addressed by the given design abstraction. More details of the Anadigm AN221E04 are provided in the next section.

2.1 The Anadigm FPAA

For detailed description of the Anadigm Vortex family of devices, see [3].

Resources: The Anadigm AN221E04 is an array of CABs (configurable analog blocks), each of which contain two opamps, 8 capacitors, a comparator, and a Successive Approximation Register (SAR) that performs 8-bit analog-to-digital conversion of signals. The device also contains one programmable lookup table that can be used to store information about the generation of arbitrary waveforms, and is shared amongst the CABs. The architecture is illustrated in the left hand block diagram of Figure 1. Any signal can be routed to the I/O pins of the device through 4 programmable I/O interface blocks and two dedicated outputs, each of which can also act as a filter or amplifier. The option for expanding the number of resources is to daisy chain multiple devices.

Table 1. Anadigm AN221E04 CAMs

CAM	CAM	CAM
Voltage Transfer Function	Inverting Differentiator	Divider
Half cycle inverting Gain Stage	Biquadratic Filter	Half Cycle Gain Stage
Half Cycle Sum/Difference Stage	DC Voltage Source	Inverting Gain Stage
Gain Stage: Switchable inputs	Bilinear Filter	Integrator
Gain Stage: Polarity Control	Half Cycle Rectifier	Half Cycle Gain Stage
Gain Stage - Output V Limiting	Inverting Sum Stage	Multiplier
Rectifier with Low Pass Filter	Sample & Hold	Sinewave Oscillator
Transimpedance Amplifier		Waveform Generator

Configurable Elements: Despite the existence of opamps and switched capacitors, the Anadigm AN221E04 does not support circuit design at this level of granularity. Instead, a circuit must be specified at the abstraction of coarser grained building blocks termed Configurable Analog Modules (CAMs) that are interconnected by wires. CAMs come predefined by Anadigm. See Table 1 for the set of available CAMs. Among the broad set is a flexible selection of filters, amplifiers and rectifiers that designers frequently use. Each CAM has programmable *options* and parameters. For example, the SumDiff CAM has a set of 4 options which decide upon clock phase, optional use of inputs 3 and 4, and inversion of each input. Its parameters are its two or more gains. To insert a CAM, the GUI must be able to fully allocate its resources from one CAB. To track how many resources are available as a circuit is defined, we reverse engineered the resource allocation strategy of the Anadigm software for GRACE.

Configuration Technology: The Anadigm FPAA uses the 'switched capacitor' technology ([10]). A switched capacitor implements an equivalent resistance by alternately opening and closing the terminals of a capacitor. Macroscopic resistance is controlled by the frequency of switching. This frequency, of course, is limited to the maximum clock frequency. Microscopic resistance is tuned by changing the capacitance value. The disadvantage of switched capacitor technology is that it performs the signal processing in discrete time domain. Thus, it requires anti-aliasing and reconstructions filters. Also, the device can only handle signals whose frequency is half its switching frequency, which is 16MHz maximum. For all blocks of the FPAA, the input and output are valid either for one of the two phases of clock or both phases. This implies a constraint on the connection of components, since a component whose output is valid on phase 1 cannot be connected to a component whose input is sampled at phase 2 of the clock and vice versa. Each internal capacitor in the Anadigm AN221E04 is drawn from a bank of capacitors. Although the software allows for the generation and routing of signals between CAMs at design time, the software only allows *dynamic* reconfiguration of the options and parameters of a circuit, not the reconfiguration of a circuit topology. The actual configuration process and mapping of the configuration bitstream is proprietary. The configuration bitstream is stored in SRAM, which is more reliable than other FPAAs based on EEPROM technology.

Configuration from GRACE: The configuration process of the Anadigm AN221E04 is proprietary. With the assistance of a colleague [11] and through a special agreement with Anadigm, we obtained a non-commerical software package that had been developed to test the GUI during product development. With this package and a Microsoft C++ compiler, GRACE sends designs from its EA module to the GUI by translating them to a series of "build commands" dispatched to the GUI. A subsequent "configure" command downloads the configuration to the device. This takes about a second which is not ideal but not prohibitive either.

While we are restricted to low to medium frequency range, we nonetheless are content. An industry segment also works in this domain due to the use of switched capacitor technology so there an industry target for whom evolutionary techniques may be useful exists.

3 GRACE: The System

GRACE is depicted in Figure 2 which shows an adaptive controller on the FPAA that controls a third order plant. The evolutionary algorithm (EA) runs on an Pentium P4 machine. It reconfigures the Anadigm AN221E04 to build new controllers, evaluate their efficiency in controlling the plant and thus guide the search to find better controllers. A summary of the components is given in Table 2.

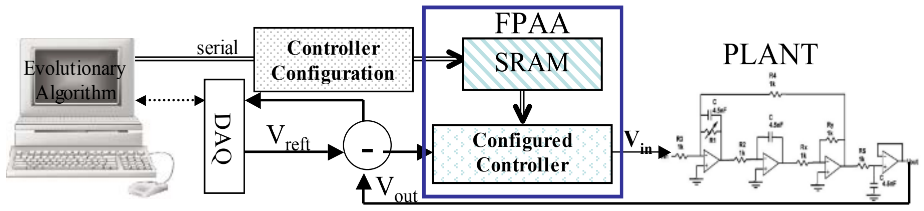


Fig. 2. GRACE Architecture

The Anadigm AN221E04 is configured by the EA via the serial port. The EA sends inputs to the hardware and extract outputs via National Instrument's PCI-6221 multifunction data acquisition card (DAQ). (The PCI-6221 DAQ board provides up to 80 analog inputs and 4 analog outputs giving GRACE scalability). The DAQ provides both analog to digital and digital to analog conversion with 16-bit resolution (for a voltage range of -10V to 10V). The reference signal to the testbench is specified by the algorithm to the DAQ as a digital waveform. The DAQ converts it to an analog signal and sends it to the testbench. Simultaneously, the DAQ converts the plant's analog output signal to a digital signal for the evolutionary algorithm to compare with the reference signal. Our system actually duplicates the reference signal sent to the controller to be matched with the plant output. This yields a time synchronized comparison between the reference and plant signals.

Table 2. GRACE: System Components, specifications, sources and cost

Component	**Specifications**	**Procured From**	**Price**
Dell Dimension 8400	3.6GHz P4 CPU, 2GB RAM	Dell Computers	\$2200 + cost of monitor
FPAA Development Kit	PCB with FPAA and 2 Signal Conditioning Dual-opamps	Anadigm	\$200
AnadigmDesigner2	Configuration Software For Win32 Platform	Anadigm	free
AutomationDoc	Documentation for Anadigm GUI Scripting	Anadigm Support	free
NI 6221 DAQ	Multifunction DAQ with analog output, PCI Card	National Instruments	\$430
NI Connect Block and Cable	Shielded Connection Block with Cable to Interface to PCI DAQ card	National Instruments	\$350

4 Choosing a Genetic Representation

The genetic representations of the evolvable hardware community have ranged from directly expressing the configuration bitstream to expressing a circuit component representation. A prominent example of the first extreme are the projects by A. Thompson [12] and his co-authors who used the Xilinx 6216. At the other is the "circuit constructing tree" which is a developmental encoding, e.g. [13]. In contrast, in GRACE a subset of the Anadigm CAM's are the functions in the sense of genetic programming. All CAMs with valid output for only one of the clock phases had their outputs connected to a "Sample and Hold" component. The GRACE genome is a cyclic graph (see Figure 3) in which each node is an instance of a CAM and directional links define the topology. A circuit has a variable number of CAMs but we implement the graph as a fixed length vector.

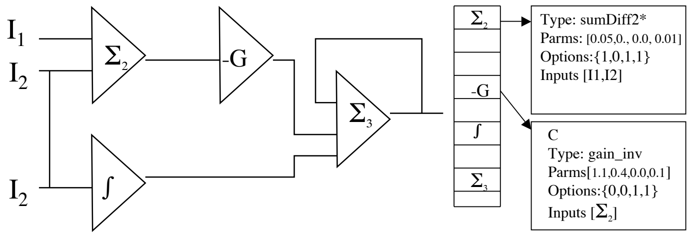


Fig. 3. Left: A circuit in GRACE is a graph. Nodes are components and edges are wires. Right: This graph is stored in a fixed length linear genome. Each object of the genome is a structure describing component, options, parameters and inputs.

Each element of the vector is a structure which specifies a CAM, its options, parameters and input source(s). Each instance of a CAM has a variable number of programmable options and parameters. For example, the SumDiff CAM has 4 options and 2 gain parameters while the simple “Half cycle Gain Stage” has only 2 options and 1 gain parameter. The genome stores in each structure another two vectors of data that the genome-to-circuit translation process interprets as parameters and options. Each vector is a fixed length. If the parameters and attributes of the CAM are fewer than the vector length, the extra values are ignored. Like the redundant nodes and links of the circuit which do not connect input to output, this redundant information is maintained in the genome.

The encoding of coarse grained components in the genome makes GRACE reminiscent of Koza’s genetic programming tree representation, e.g. in [14]. The obvious difference is the cyclic graph versus the tree. Another difference is the genome length – fixed in GRACE’s case and variable in Koza’s. The physical limitation of resource quantities on the device demand that GRACE not evolve a genome that requires more resources than on the device. This is ensured by the fixed length genome and by the decoding algorithm that maps the genome to a series of build commands. The decoding algorithm makes use of a resource manager to account for resources that will be used on the device as it translates the genome into “build” commands. If it ever encounters a CAM (i.e. node) for which the resources cannot be allocated, it replaces this node with a wire. GRACE’s genome is also influenced by Miller’s Cartesian Genetic Programming (CGP), [15]. The CGP genome is also a graph mapped to a matrix of varying component with links between and among columns.

The search algorithms: We use the standard generation based processing loop of an EA to conduct topology search. At initialization, a population of random genomes is created. Each genome is mapped to a circuit topology with each instance of a CAM specified using its input list, options and parameter values. Serially each genome is configured on the device and given a test signal. The resulting output signal is captured and evaluated in comparison to a desired output signal. The error is mapped to a genome fitness. After the entire current generation is tested, tournament selection supplies parents for the next generation. Each parent is copied to create an offspring in the next generation. Offspring are mutated before being added to the population of the next generation. Mutation can be applied in two ways to the genome: to a CAM instance by changing its type and, to the input(s) of a CAM by changing a link in the graph.

Given a topology, finding the parameters of the CAM is a numerical optimization problem. Recently, Particle Swarm Optimization (PSO) has emerged as an efficient approach to do fast numerical optimization to find the global optimum [16]. We use PSO to set the parameters of CAMs rather than evolving it together with the topology by the EA. We believe that performing the steps of topology search and component optimization separately makes the problem more tractable for the EA. These two steps of topology search using an EA and component optimization using PSO can be combined in various ways which shall effect the efficiency and speed of the algorithm. For the current set of experiments, we run PSO on each individual in the EA population and assign the best of swarm fitness to the individual. Intuitively, this approach assigns the topology fitness according to its best performance given the most suitable parameter values. Other approaches which trade speed for efficiency and vice-versa are under study.

5 GRACE in Action: Evolving a Controller

We have initially used GRACE to evolve a controller for the simple first order plant shown in Figure 4 (left). The plant has bandwidth of 338.6Hz and a steady state gain of 2.5. The CAMs in the primitive set for the given problem can be found in Table 3 along with their respective parameters and options. These CAMs are capable of creating any transfer function (realizable given the capacity of the FPAA) including the ubiquitous Proportional-Integral-Differential control. The population size was 15 with tournament selection of size 3 and elitism for 3 individuals. A run was 10 generations with the probability of mutating a CAM 0.45 and a wire 0.45. The PSO ran 6 iterations every generation on every individual with a swarm size of 4.

Fitness Function: Though a simple step function would seem to be all that is required to evaluate a controller, we used a more complex signal to ensure that GRACE did not evolve a signal generator regardless of the input. The signal and a candidate circuit's response is shown on the right in Figure 4. The signal has six voltage levels (-1.5V, -0.75V, -0.375, 0.375, 0.75V. and 1.5V) and changes state every 4.16ms. We sampled the signal at 125 KHz. The fitness of a circuit is the weighted sum of squared errors between the circuit's output signal and the test signal. The fitness function weights can be tuned to trade-off criteria of settling time, peak overshoot and steady state error. For instance, more weight to the error in latter part of the step response shall bias the search towards controllers with lower steady state error and shall care less for rise time and peak overshoot. For the current set of experiments, we used the time-weighted least squares, which increases the weights linearly with time. It is postulated in [17], such a fitness function is ideal for judging the efficiency of a controller.

Table 3. CAMs used in the GRACE Function Set for Controller Evolution. Asterisk indicates output is connected to sample and hold block for two clock phase results.

CAM	**Parameter(s)**	**# In**
SumDiff-2*	inputs gain value(s)	2
SumDiff-3*	inputs gain value(s)	3
Inverting Differentiator	diff. constant (us)	1
Integrator	gain	1
Gain Inverter	gain	1
Gain*	gain	1
Wire	0	1

Evolved Solutions: The system evolved solutions with high fitness value (validated by visual inspection of generated waveforms) that instantiate various control strategies, for instance, proportional control, integral control or lossy integral control. Evolution also found interesting ways to build solutions, like use of a differentiator in feedback to evolve a lossy integrator and using multiple feedback to realize different gains (including high gain through positive feedback) for proportional control.

Analysis of one of the best-of-run controller showed how evolution can *think* out-of-the-box. Figure 5 shows the controller as seen in the Anadigm GUI on the left and the equivalent simple block diagram on the right. Simple hand-analysis shows that the solution is a filter. The summing-integrator filter topology is a well-known approach to synthesize filters. It is counter-intuitive why a filter would be a good controller. Evolution exploits the high integration-constant (of the order of Mega per second) realizable by the integrator. It evolves a high gain filter with a large bandwidth that has an integrator in both the forward and feedback paths. This effectively behaves like proportional control with a large gain. The high gain of the P-control reduces the steady-state error thus contributing to high fitness. This solution has not been included in discussion for its usefulness in a real scenario, but due to illustrate the ability of algorithm to

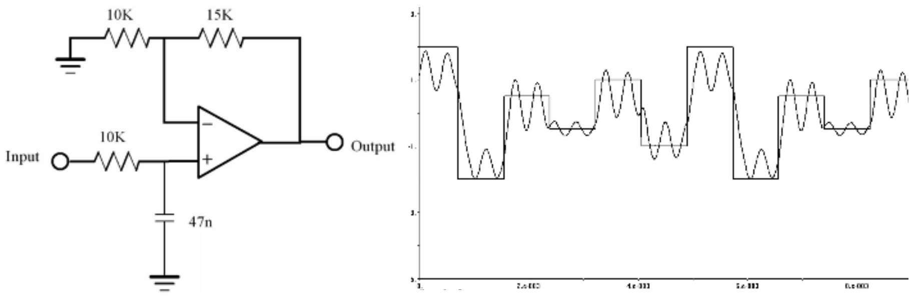


Fig. 4. Left: Plant for evolved controller, Right: Fitness function test signal (square wave) with example circuit's output signal for the controller experiment

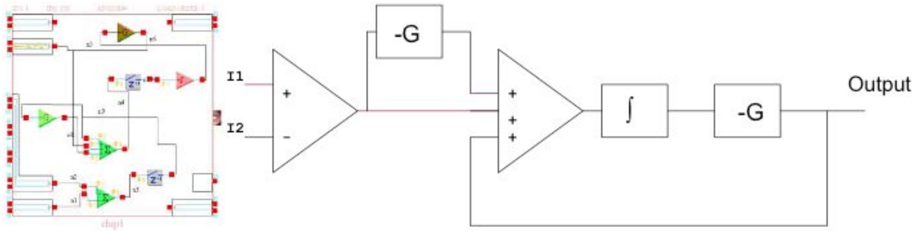


Fig. 5. Left: An evolved filter solution displayed from GUI , Right: Schematic of same solution

synthesize interesting topologies and the capability of evolution to explore realms of unconventional design even when it uses coarse-grained building blocks.

Work is underway to study the solutions generated and use a carefully crafted fitness function to better capture the characteristics of the controller. With an effective fitness function instantiated in the system, we shall determine the usefulness of the circuits evolved and compare them with those evolved on other platforms such as the FPTAs. We also plan to study how variation in the evolutionary algorithm (method/parameters) affects its ability to search for a solution in the given problem domain.

6 Summary

By combining the exploitation of coarse grained elements with intrinsic testing on a COTS device, we think GRACE comprises a distinctive approach to analog EHW. This paper's goal has been to elucidate our decision process in engineering GRACE. We feel our decision to use the Anadigm AN221E04 forges GRACE's identity. It is a COTS rather than custom device. The proprietary nature of its configuration process can be circumvented for practicality. It uses SRAM to hold a configuration. This makes it suited as a component of an adaptive, fault tolerant system. It exploits switched capacitor technology. This allows its evolved designs to conform with industry specifications and be realizable. This will facilitate the ultimate placement of evolved circuits in the field.

Finally, in using the Anadigm AN221E04, it offers coarse grained elements. Coarse granularity makes GRACE contrast with FPTA approaches by exchanging flexibility with higher level building block abstraction. We think that GRACE enables a parallel set of investigations that will provide interesting comparisons between the non-linear design space of the FPTA and the human oriented, conventional design space. We believe our choices additionally provide us with traction into both adaptive, robust hardware evolution and the more traditional pursuit of analog CAD. This will be the direction of our future research using GRACE.

Acknowledgements

We would like to thank Anadigm, Dimitri Berensen, Garrison Greenwood, David Hunter, Didier Keymeulen, Adrian Stoica, and Eduardo Torres-Jara for their contributions to the development of GRACE.

References

1. Stoica, A., Zebulum, R.S., Keymeulen, D.: Progress and challenges in building evolvable devices. In: Evolvable Hardware. (2001) 33–35
2. Greenwood, G., Hunter, D.: Fault recovery in linear systems via intrinsic evolution. In Zebulum, R., Gwaltney, D., Hornby, G., Keymeulen, D., Lohn, J., Stoica, A., eds.: Proceedings of the NASA/DoD Conference on Evolvable Hardware, Seattle, Washington, IEEE Computer Society (2004) 115–122

3. Anadigm: AN221E04 datasheet: Dynamically reconfigurable fpaa. http://www.anadigm.com/doc/DS030100-U006.pdf (2004)
4. Stoica, A., Zebulum, R., Keymeulen, D., Tawel, R., Daud, T., Thakoor, A.: Reconfigurable vlsi architectures for evolvable hardware: from experimental field programmable transistor arrays to evolution-oriented chips. IEEE Transactions on VLSI Systems, Special Issue on Reconfigurable and Adaptive VLSI Systems **9**(1) (2001) 227–232
5. Stoica, A., Zebulum, R.S., Keymeulen, D.: Polymorphic electronics. In Liu, Y., Tanaka, K., Iwata, M., Higuchi, T., Yasunaga, M., eds.: ICES. Volume 2210 of Lecture Notes in Computer Science., Springer (2001) 291–302
6. Keymeulen, D., Stoica, A., Zebulum, R.: Fault-tolerant evolvable hardware using field programmable transistor arrays. IEEE Transactions on Reliability, Special Issue on Fault-Tolerant VLSI Systems **49**(3) (2000) 305–316
7. Stoica, A., Keymeulen, D., Zebulum, R.S.: Evolvable hardware solutions for extreme temperature electronics. In: Evolvable Hardware, IEEE Computer Society (2001) 93–97
8. Langeheine, J., Becker, J., Fölling, S., Meier, K., Schemmel, J.: Initial studies of a new vlsi field programmable transistor array. In Liu, Y., Tanaka, K., Iwata, M., Higuchi, T., Yasunaga, M., eds.: Evolvable Systems: From Biology to Hardware: Proceedings of 4th International Conference, ICES 2001. Volume 2210 of Lecture Notes in Computer Science., Tokyo, Japan, Springer-Verlag (2001)
9. Stoica, A., Zebulum, R., Ferguson, M., Keymeulen, D., Duong, V.: Evolving circuits in seconds: Experiments with a stand-along board-level evolvable system. In Stoica, A., Lohn, J., Katz, R., Keymeulen, D., Zebulum, R.S., eds.: Proceedings of the NASA/DoD Conferenece on Evolvable Hardware, Alexandria, Virginia, IEEE Computer Society (2002) 129–130
10. Allen, P.E., Sanchez-Sinencio, E.: Switched Capacitor Circuits. VanNostrand Reinhold Company (1984)
11. Berenson, D.: Personal communication. email: January 18, 2005 (2005)
12. Thompson, A.: Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution. Springer-Verlag (1998)
13. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Four problems for which a computer program evolved by genetic programming is competitive with human performance. In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation. Volume 1., IEEE Press (1996) 1–10
14. Koza, J.R., Kean, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV:Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers (2003)
15. Miller, J.F., Thompson, P.: Cartesian genetic programming. In: Proceedings of Third European Conference on Genetic Programming, Springer-Verlag (2000) 121–132
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the Fourth IEEE International Conference on Neural Networks, IEEE Press (1995)
17. Krohling, R.A., Jaschek, H., Rey, J.: Designing PI/PID controllers for a motion control system based on genetic algorithms. In: Proceedings of the 12th IEEE International Symposium on Intelligent Control. (1997) 125–130

On the Practical Limits of the Evolutionary Digital Filter Design at the Gate Level

Lukáš Sekanina and Zdeněk Vašíček

Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, Brno 612 66, Czech Republic
sekanina@fit.vutbr.cz, xvasic11@stud.fit.vutbr.cz

Abstract. Simple digital FIR filters have recently been evolved directly in the reconfigurable gate array, ignoring thus a classical method based on multiply–and–accumulate structures. This work indicates that the method is very problematic. In this paper, the gate-level approach is extended to IIR filters, a new approach is proposed to the fitness calculation based on the impulse response evaluation and a comparison is performed between the evolutionary FIR filter design utilizing a full set and a reduced set of gates. The objective of these experiments is to show that the evolutionary design of digital filters at the gate level does not produce filters that are useful in practice when linearity of filters is not guaranteed by the evolutionary design method.

1 Introduction

FIR (finite impulse response) filters and IIR (infinite impulse response) filters represent two important classes of digital filters that are utilized in many applications. For these filters, a rich theoretical understanding as well as practical design experience have been gained in the recent decades [6]. Typically, their implementation is based on *multiply–and–accumulate* structures (regardless on software or hardware implementation). Alternative design paradigms (such us multiplierless designs) have also been formulated [7].

With the development of real-world applications of evolutionary algorithms, researchers have started to evolve digital filters. Miller has introduced probably the most radical idea for their design [9, 10, 11]: In evolutionary design process, target filters are composed from elementary gates, ignoring thus completely the well-developed techniques based on multiply–and–accumulate structures. The main practical potential innovation of this approach could be that the evolved filters are extremely area-efficient in comparison with the standard approach. We should understand this approach as a demonstration that the evolution is capable to put some gates together in order to perform a very simple filtering task. Definitely, the approach is not able to compete against the standard methods. A similar approach has been adopted for functional level evolution of IIR filters [3].

In contrast to an optimistic view presented in the mentioned papers, this work indicates that the approach is very problematic. In this paper, Miller's gate-level approach is extended to IIR filters, a new approach to the fitness calculation is proposed based on the impulse response evaluation and a comparison is performed between the evolutionary FIR filter design utilizing a full set of gates and reduced set of gates. The objective of these experiments is to support the following hypothesis: "Evolutionary design of digital filters at the gate level does not produce filters that are useful in practice when linearity of filters is not guaranteed." Two approaches that could ensure the linear behavior of evolved filters will be discussed.

The rest of the paper is organized as follows. Section 2 introduces the area of digital filter design and the use of evolutionary techniques in this area. In Section 3, the proposed approach is described to the evolutionary design of FIR and IIR filters. Results of experiments are reported in Section 4. Section 5 discusses the advantages and disadvantages of the proposed approach. Conclusions are given in Section 6.

2 Conventional and Evolutionary Design of Digital Filters

A *discrete-time* system is essentially a mathematical algorithm that takes an input sequence, $x(n)$, and produces an output sequence, $y(n)$ [6]. A *digital filter* is an example of discrete-time system. A discrete-time system may be linear or nonlinear, time invariant or time varying. *Linear time-invariant* (LTI) systems form an important class of systems used in digital signal processing.

A discrete-time system is said to be linear if it obeys the principle of *superposition.* Consider that $x_1(n)$ and $x_2(n)$ are two input signals and $y_1(n)$ and $y_2(n)$ are corresponding responses of the filter. The filter is *linear* if the following holds:

$$a_1x_1(n) + a_2x_2(n) \rightarrow a_1y_1(n) + a_2y_2(n) \tag{1}$$

where a_1 and a_2 are arbitrary constants.

A discrete-time system is said to be *time-invariant* if its output is independent of the time the input is applied, i.e. a delay in the input causes a delay by the same amount in the output signal.

The input–output relationship of an LTI system is given by the convolution sum

$$y(n) = \sum_{k=-\infty}^{+\infty} h(k)x(n-k) \tag{2}$$

where $h(k)$ is the *impulse response* of the system. The values of $h(k)$ completely define the discrete-time system in the time domain.

A general IIR (infinite impulse response) digital filter is described by equation

$$y(n) = \sum_{k=0}^{N} b_k x(n-k) - \sum_{k=1}^{M} a_k y(n-k). \tag{3}$$

The output samples $y(n)$ are derived from current and past input samples $x(n)$ as well as from current and past output samples. Designer's task is to propose values of coefficients a_k and b_k and size of vectors N and M. In FIR (finite impulse response) filters, the current output sample, $y(n)$, is a function only of past and present values of the input, i.e.

$$y(n) = \sum_{k=0}^{N} b_k x(n-k). \tag{4}$$

The stability and linear phase are main advantages of FIR filters. On the other hand, in order to get a really good filter many coefficients have to be considered in contrast to IIR filters. In general, IIR filters are not stable (because of feedback). FIR filters are algebraically more difficult to synthesize.

Various methods have been proposed to design digital filters (such as frequency sampling method and window method for FIR filters and pole/zero placement and bilinear z-transform for IIR filters). These methods are well developed and represent an approach to digital filter design widely adopted by industry. Digital filters are usually implemented either on DSPs or as custom circuits. Their implementation is based on multipliers and adders. The quality of output signal, speed of operations and cost of hardware implementation are important factors in the design of digital filters. The multiplier is the primary performance bottleneck when implementing filters in hardware as it is costly in terms of area, power and signal delay. Hence multiplierless filters were introduced in which multiplication is reduced to a series of bitshifts, additions and subtractions [12, 7].

Evolutionary algorithms have been utilized either to optimize filter coefficients [4] or to design a complete filter from chosen components. In particular, structures of multiplierless filters were sought by many authors [12, 5, 1]. As these filters are typically composed of adders, subtracters and shifters (implementing multiplication/division by the powers of two) they exhibit "linear" behavior for the required inputs.

Miller has pioneered the evolutionary approach in which FIR filters are constructed from logic gates [9, 10, 11]. He has used an array of programmable gates to evolve simple low-pass, high-pass and band-pass filters that are able to filter simple sine waves and their compositions. The unique feature of these filters is that they are composed of a few tens of gates; thus reducing the implementation costs significantly in comparison with other approaches. The evolved filters do not work perfectly and they are far from the practical use; however, Miller has demonstrated that *quasi-linear* behavior can be obtained for some particular problems. The gate arrays are carrying out filtering without directly implementing a *difference equation* – an abstract model utilized for filter design. The fitness function can be constructed either in the frequency domain or time domain. In both cases Miller has obtained similar results. However, he mentioned that: "Experience suggests that gate arrays that are evolved using a fitness function which looks at the frequency spectrum of the circuit output appear to be more linear in behavior than using an error based measure of fitness" [10].

Recently Gwaltney and Dutton have utilized similar approach to evolve IIR filters at the functional level (for 16bit data samples) [3]. Their filters are composed of adders, multipliers and some logic functions; therefore, they are non-linear. They have evolved low-pass IIR filters using a couple of components. The filter fails to function properly when the input is changed to a signal that is "significantly" different from that used during evolution.

3 Gate-Level Evolution of Digital Filters

Similarly to Miller [9, 10, 11], Cartesian genetic programming (CGP) is utilized in this paper to evolve simple digital filters at the gate level. Although all internal data are at 1 bit and gates perform elementary logic operations, the inputs as well as outputs are interpreted as 8 bit values.

3.1 Cartesian Genetic Programming

CGP models a reconfigurable circuit, in which digital circuits are evolved, as an array of u (columns) $\times$ v (rows) of programmable elements (gates) [8]. The number of circuit inputs n_i and outputs n_o are fixed. Each gate input can be connected to the output of some gate placed in the previous columns or to some of the circuit inputs. L-back parameter defines the level of connectivity and thus reduces/extends the search space. For example, if L=1 only neighboring columns may be connected; if L=u, the full connectivity is enabled. A circuit configuration is defined using $3.u.v + n_o$ integers: the three integers describe the connection and function of a single gate and n_o integers specify the connection of outputs. Every gate performs one of functions specified in function set F. Figure 1 provides an example.

Miller has originally used a very simple variant of evolutionary algorithm to produce configurations for the programmable circuit [8]. Our algorithm is very similar. It operates with the population of 5 individuals; every new population consists of mutants of the best individual. Only the mutation operator has been utilized that modifies 1–3 randomly selected genes of an individual.

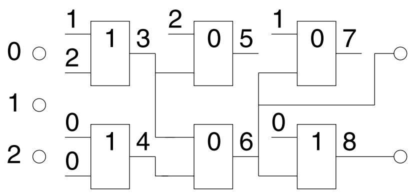


Fig. 1. An example of a 3-input circuit. CGP parameters are as follows: L = 3, u = 3, v = 2, F = {AND (0), OR (1)}. Gates 5 and 7 are not utilized. Chromosome: 1,2,1, 0,0,1, 2,3,0, 3,4,0 1,6,0, 0,6,1, 6, 8. The last two integers indicate the outputs of the circuit.

3.2 Gate-Level Digital Filters and CGP

Figure 2a shows our modification of CGP to design FIR filters. A delay chain was created using two registers. We can observe that the three w-bit samples are processed by the gate array ($w = 8$). Before simulation is started, delay registers are cleared. The following operations are repeated N-times (N is the number of samples): the ith sample is right-shifted by w bits and the $(i+1)$ sample possess its position. Then the output value is calculated and interpreted as an integer value.

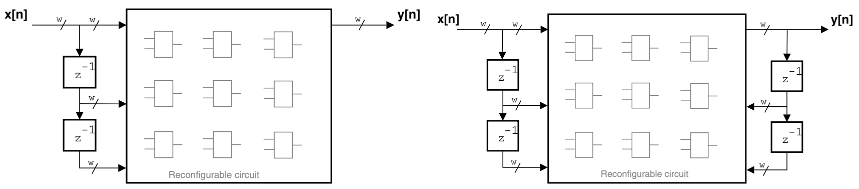
Figure 2b shows the approach utilized to evolve IIR filters using CGP. In addition to the previous approach, the output delay registers have to be shifted to send the obtained output value back to the circuit. Because of the feedback, the IIR filter simulation is much slower than the FIR filter simulation.

The proposed fitness function works in the time domain. The objective is to minimize the difference between measured signal $y(n)$ and target signal $y_{ref}(n)$, i.e.

$$fitness_{MSE} = -\sqrt{\sum_{i=0}^{N-1} (y(i) - y_{ref}(i))^2}. \tag{5}$$

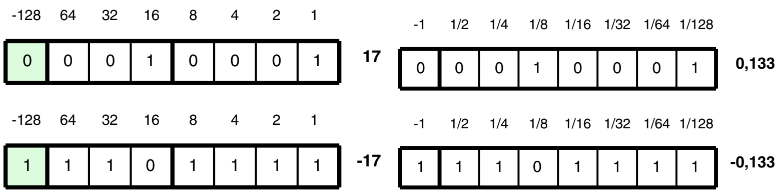
The main problem is to determine which signals should be included into the training set. Ideally, all frequencies and shapes should be testes; however, it is not tractable.

An alternative approach could be to apply the unit impulse (i.e. the signal that contains all frequencies) at the input and to measure the impulse response. This



(a) (b)

Fig. 2. CGP utilized to design (a) FIR filters and (b) IIR filters



(a) (b)

Fig. 3. Comparison of 2's complement encoding (a) and fraction arithmetic (b)

approach will be utilized to design IIR filters. The role of CGP is to find such the filter whose impulse response is as close as possible to the required impulse response. We have to work with real numbers because the values of impulse responses range from –1 to +1. In our case we will represent real numbers in the fraction arithmetic (which is based on 2's complement encoding as shown in Figure 3).

4 Results

The following CGP parameters represent the basic setup for experiments: $u = 15, v = 15, L = 15, n_i = 24, n_o = 8$, population size 5, 15 million generations, function set: $F = \{c = a, c = a$ and $b, c = a$ or $b, c = a$ xor $b, c =$ not $a, c =$ not $b, c = a$ and (not b)$, c = a$ nand $b, c = a$ nor $b\}$. CGP was implemented in C++. The evolved filters were analyzed using Matlab.

4.1 Low-Pass Filter I

The training input signal consists of composition of frequencies f_1 and $f_3 = 3f_1$. As the circuit should carry out low-pass filter, the output should contain f_1 only (which will be expressed in this paper as: $f_1 + f_3 \rightarrow f_1$). In particular, we utilized $N = 128$ samples and $x_1(n) = 127 + 100.\sin(2\pi n/128)$ and $x_3(n) = 127 + 100.\sin(2\pi.3n/128)$.

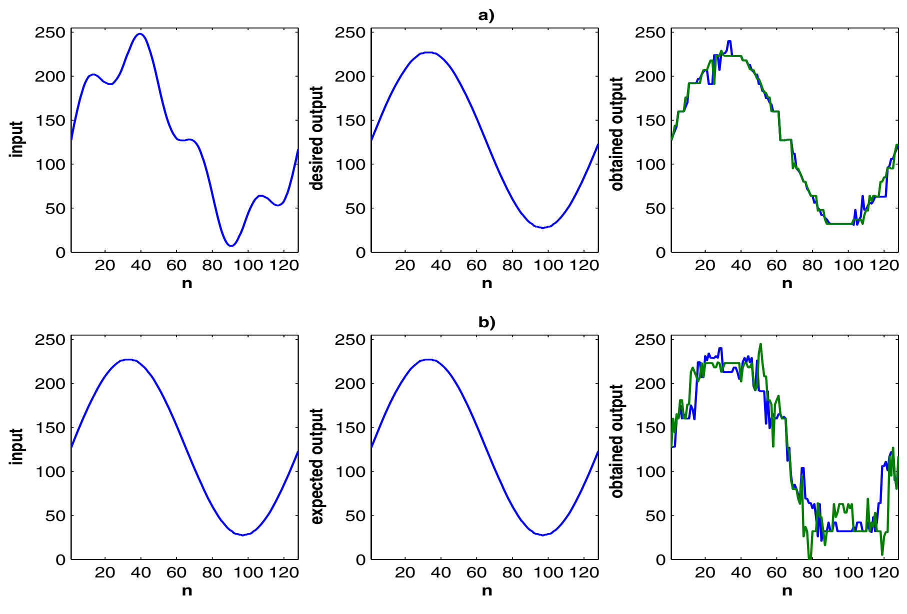


Fig. 4. Behavior of the evolved low-pass filter: input signal (left), required output signal (center), output signal (right); (a) training signal, (b) test signal

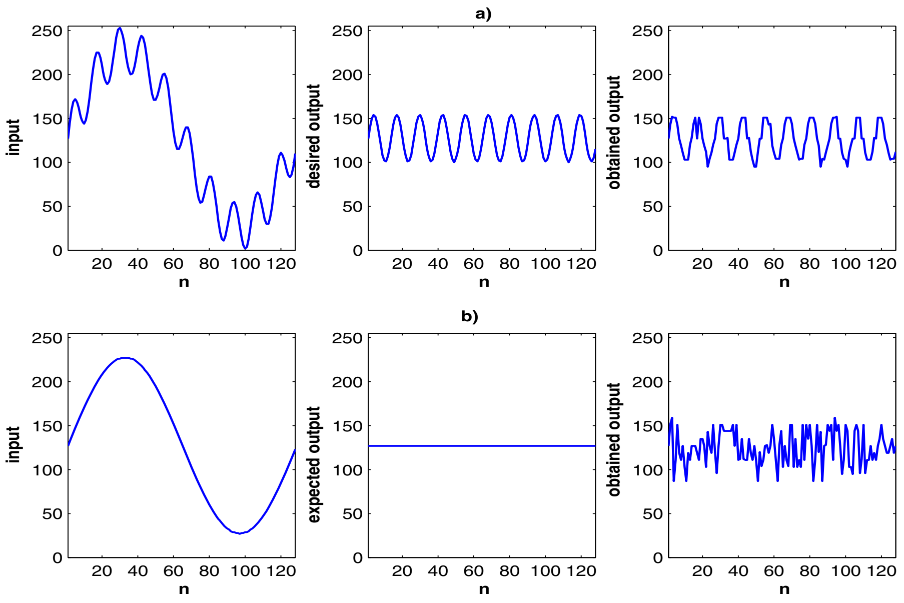


Fig. 5. Behavior of the evolved high-pass filter: input signal (left), required output signal (center), output signal (right); (a) training signal, (b) test signal

Figure 4 shows the behavior of the best evolved filter. We utilized the signal with f_1 as a test signal and observed that the evolved circuit modifies the signal although it should transmit the signal without any change. Therefore, the circuit cannot be understood as a perfect filter.

4.2 High-Pass Filters

Figure 5 shows behavior of an evolved high-pass filter whose function was specified as $f_1 + f_{10} \rightarrow f_{10}$. Although the result for the training signal seems to be correct, the filter does not work for other signals at all.

4.3 Low-Pass Filter II

In order to evolve more robust filters, we included more requirements to the fitness function. A low-pass filter was specified as: $f_1 + f_3 \rightarrow f_1$, $f_1 + f_5 \rightarrow f_1$ and $f_5 \rightarrow f_0$. The evolved filter exhibits an acceptable (but not perfect) behavior for training signals as well as test signals (see Figure 6). This approach could eventually be utilized to design an extremely cheap filter which is supposed to work for a limited amount of input signals.

4.4 The Impulse Response in Fitness Calculation

In this experiment, we defined the required impulse response and were interested whether CGP is able to find an IIR filter with the same impulse response. Figure

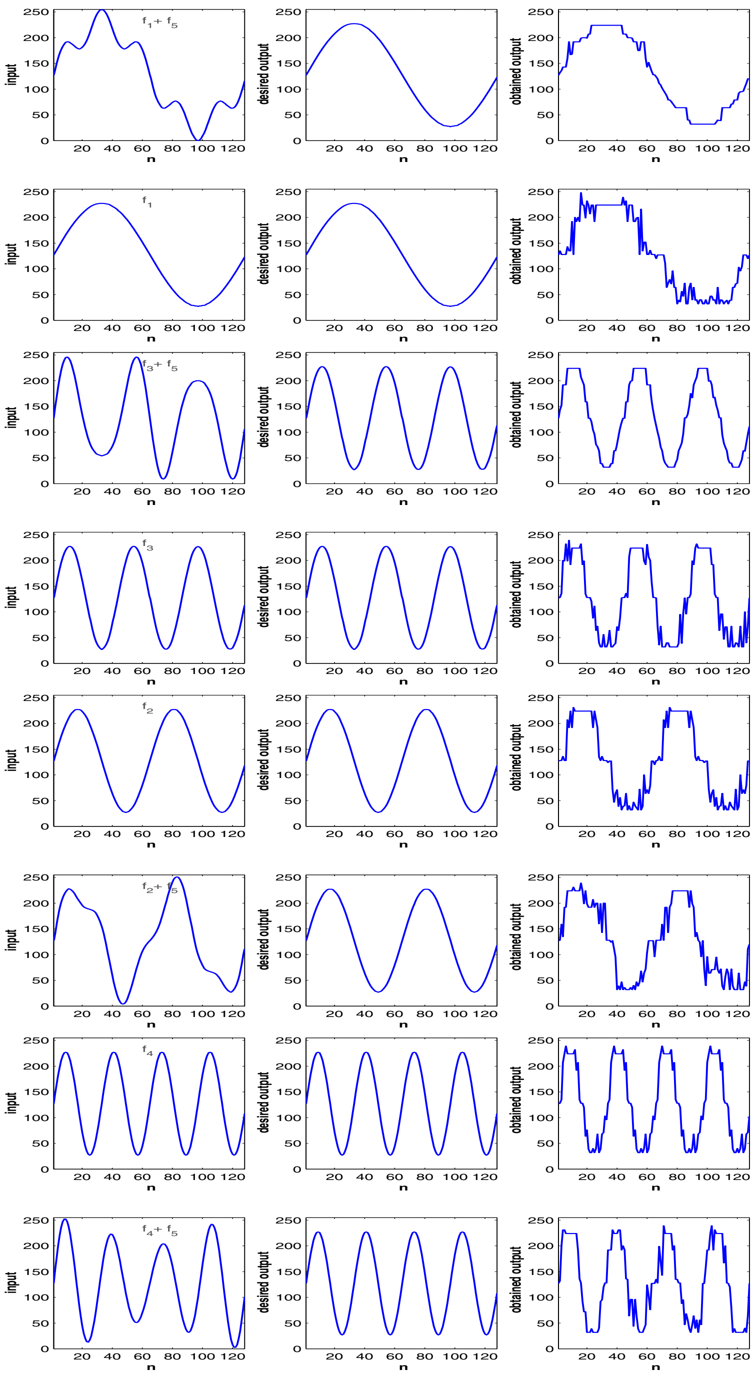
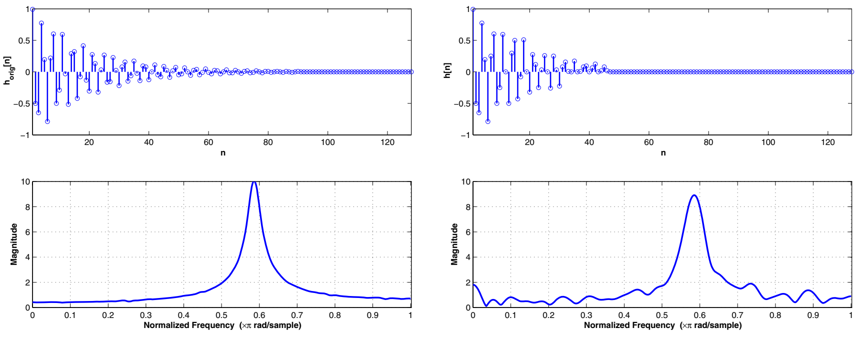


Fig. 6. Behavior of the evolved low-pass filter II: input signal (left), required output signal (center), output signal (right)



(a) (b)

Fig. 7. The impulse response: (a) required (b) evolved

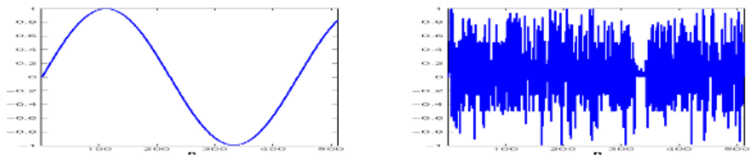


Fig. 8. The input signal (left) and the "filtered" signal obtained using the evolved filter (right)

7a and 7b show that it is possible and furthermore, the corresponding frequency characteristic is also very close to the perfect one.

Unfortunately, the evolved circuit is not a filter at all. Figure 8 shows its response to a simple sine input signal. The output signal should exhibit a very small amplitude; however, its behavior is completely random.

4.5 A Reduced Set of Gates

In cellular automata and other circuits, a "linear" variant of the model is sometimes introduced (see, for example, linear cellular automata [2]). Then, logic circuits are composed using the gates *xor* and *not* because their analysis is amenable to algebraic methods. In order to explore whether this type of linearity could be useful for gate-level filters, we arranged the following experiments. The objective was to evolve a high-pass FIR filter specified as $f_1 + f_{10} \rightarrow f_{10}, f_3 + f_{10} \rightarrow f_{10}, f_5 + f_{10} \rightarrow f_5 + f_{10}, f_{10} \rightarrow f_{10}$. We utilized CGP with $u = 15$ and $v = 15$ and produced 20 millions of generations. The evolved filter was tested using signals: $f_{10}, f_1, f_3, f_4, f_5, f_0$. We have considered two scenarios: (1) a complete set of gates, F, and (2) a reduced set of gates, F', containing {xor, not}. Table 1 summarizes the obtained results. We can observe that the circuits composed of *xor* and *not* gates (scenario 2) are much smaller and more general than those obtained in case (1). On the other hand, the filter evolved using a complete set

Table 1. Filters evolved using the complete set of gates and reduced set of gates

Filter	MSE (training data)	MSE (test data)	# of used gates
Complete set (scenario 1)	132,583	738,075	197
Reduced set (scenario 2)	369,275	525,700	44

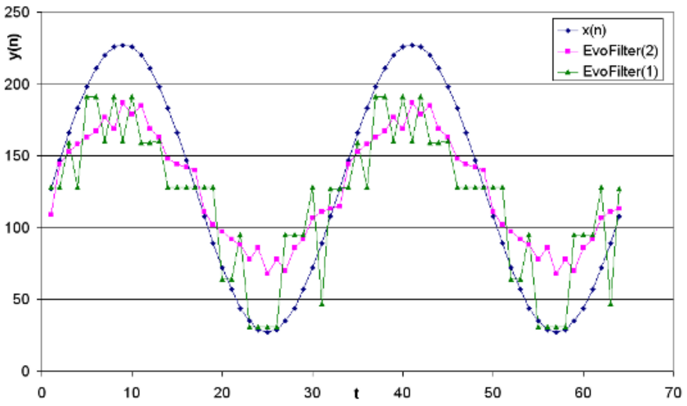


Fig. 9. Signal $f_4 = x(n)$: The outputs of filters evolved with the compelte set of gates (EvoFilter 1) and reduced set of gates (EvoFilter 2)

of gates is more adapted to training signals. However, Figure 9 shows that the output response is acceptable neither for scenario 1 nor 2.

5 Discussion

The common result of experiments performed herein, by Miller [9, 10, 11] and by Gwaltney and Dutton [3] is that the evolved filters do not work when they are required to filter signals *different* from training signals. Moreover, the evolved filters do not generate perfect responses either for training signals. The evolved circuits are not, in fact, filters. In most cases they are combinational circuits trained on some data that are not able to generalize. In order to obtain real filters, the design process must guarantee that the evolved circuits are *linear*. There are two ways how to ensure that: (1) The circuit is composed of components that are linear and the process of composition always ensures a linear behavoir. This approach is adopted by many researchers (e.g., [1]) but not by the methods discussed in this paper. (2) Linearity is evaluated in the fitness calculation. Unfortunately, that is practically impossible because all possible input signals should be considered, which is intractable. Note that Miller's fitness function [11] has promoted the filters exhibiting the *quasi-linear behavior*; however, it does not guarantee (in principle) that a candidate filter is linear although the filter has obtained a maximum fitness score.

In this paper, we have evolved FIR as well as IIR "filters" and proposed the approaches based on the impulse response and the reduced set of (linear) gates. However, none of them have led to satisfactory results. On the basis of experiments performed in this paper and results presented in [9, 10, 11], we are claiming that the gate level evolutionary design of digital filters is not able to produce filters "useful" in practice if the linearity is not guaranteed by the evolutionary design process. As we do not know now how to ensure the linear behavior, the approach should be considered as curious if one is going to design a digital filter.

There could be some benefits coming with this "unconventional" filter design. It was shown that circuits can be evolved to perform filtering task when sufficient resources are not available (e.g. a part of chip is damaged) [5] or when some noise in presented in input signals [10]. Furthermore, as Miller has noted, "The origin of the quasi-linearity is at present quite mysterious. ... Currently there is no known mathematical way of designing filters directly at this level." Possibly we could discover novel design principles by analyzing the evolved circuits.

The evolutionary design is very time consuming. In order to produce 20 millions of generations with a five-member population, the evolutionary design requires 29.5 hours for IIR filter and 6 hours for FIR filter (on a 2.8 GHz processor).

6 Conclusions

In this paper, the gate-level approach to the digital filter design was extended to IIR filters, a new approach was proposed to the fitness calculation based on the impulse response evaluation and a comparison was performed between the full set of gates and reduced set of gates for the evolutionary FIR filter design. On the basis of experiments performed herein and the results presented in literature we have recognized that the gate level evolutionary design of digital filters is not able to produce "real" filters. Therefore, this approach remains a curiosity rather than a design practice.

Acknowledgment

The research was performed with the Grant Agency of the Czech Republic under No. 102/04/0737 *Modern Methods of Digital Systems Synthesis.*

References

[1] Erba, M., et al.: An Evolutionary Approach to Automatic Generation of VHDL Code for Low-Power Digital Filters. In: Proc. of the 4th European Conference on Genetic Programming, LNCS 2038, Springer-Verlag, 2001, p. 36–50

[2] Ganguly, N. et al. A survey on cellular automata. Technical report, Centre for High Performance Computing, Dresden University of Technology, December 2003

[3] Gwaltney, D., Dutton, K.: A VHDL Core for Intrinsic Evolution of Discrete Time Filters with Signal Feedback. In Proc. of 2005 NASA/DoD Conference on Evolvable Hardware, IEEE Comp. Society Press, 2005, p. 43–50

[4] Harris, S. P., Ifeachor, E. C.: Automating IIR filter design by genetic algorithm. Proc. of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), No. 414, IEE, London, 1995, p. 271–275

[5] Hounsell, B. I., Arslan, T., Thomson, R.: Evolutionary design and adaptation of high performance digital filters within an embedded reconfigurable fault tolerant hardware platform. Soft Computing. Vol. 8, No. 5, 2004, p. 307–317

[6] Ifeachor, E. C., Jervis, B. W.: Digital Signal Processing: A Practical Approach, 2nd edition, Pearson Education, 2002

[7] Martinez-Peiro, M., Boemo, E. I., Wanhammar, L.: Design of High-Speed Multiplierless Filters Using a Nonrecursive Signed Common Subexpression Algorithm. IEEE Trans. Circuits Syst. II, Vol. 49, No. 3, 2002, p. 196-203

[8] Miller, J., Job, D., Vassilev, V.: Principles in the Evolutionary Design of Digital Circuits – Part I. Genetic Programming and Evolvable Machines, Vol. 1, No. 1, 2000, p. 8–35

[9] Miller, J.: Evolution of Digital Filters Using a Gate Array Model. In: Proc. of the Evolutionary Image Analysis, Signal Processing and Telecommunications Workshop. LNCS 1596, Springer-Verlag, 1999, p. 121–132

[10] Miller, J.: On the filtering properties of evolved gate arrays. In Proc. of NASA/DOD Workshop on Evolvable Hardware, IEEE Comp. Society, 1999, p. 2–11

[11] Miller, J.: Digital Filter Design at Gate-level using Evolutionary Algorithms. In Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99). Morgan Kaufmann, 1999, p. 1127–1134

[12] Wade, G., Roberts, A., Williams, G.: Multiplier-less FIR filter design using a genetic algorithm. IEE Proceedings in Vision, Image and Signal Processing, Vol. 141, No. 3, 1994, p. 175–180

Image Space Colonization Algorithm

Leonardo Bocchi[1] and Lucia Ballerini[2]

[1] Dept. of Electronics and Telecommunications, University of Florence, Via S.Marta 3, Firenze 50139, Italy
leo@asp.det.unifi.it
[2] Dept. of Food Science, Swedish University of Agricultural Sciences, P.O. Box 7051, Uppsala, 75007, Sweden
lucia@cb.uu.se

Abstract. This paper describes an image segmentation method based on an evolutionary approach. Unlike other application of evolutionary algorithms to this problem, our method does not require the definition of a global fitness function. Instead a survival probability for each individual guides the progress of the algorithm. The evolution involves the colonization of a bidimensional world by a number of populations. The individuals, belonging to different populations, compete to occupy all the available space and adapt to the local environmental characteristics of the world. We present various sets of experiments on simulated MR brain images in order to determine the optimal parameter settings. Experimental results on real image are also reported. Images used in this work are color camera photographs of beef meat.

1 Introduction

Image segmentation plays an important role in image processing, and it is usually the starting point for any subsequent analysis. The goal of image segmentation is to partition an image into homogeneous regions according to various criteria such as for example gray level, color, or texture. Image segmentation has been the subject of intensive research, and a wide variety of techniques have been reported in literature. A good review of these methods can be found in [1].

Alternative approaches to exploit the metaphor of natural evolution in the context of image segmentation have been proposed. The genetic learning system proposed by Bhanu et al. [2] allows the segmentation process to adapt to image characteristics, which are affected by varying environmental factors such as the time of the day, condition on cloudiness, etc. Bhandarkar and Zhang [3] use the genetic algorithm to minimize the cost function that is used to evaluate the segmentation results. Andrey [4] describes a selectionist relaxation algorithm, whereby the segmentation of an input image is achieved by a population of elementary units iteratively evolving through a fine-grained distributed genetic algorithm. Liu and Tang [5] present an autonomous agent-based approach, where a digital image is viewed as a two-dimensional cellular environment in which the agents inhabit and attempt to label homogeneous segments. Veenman et al. [6]

use a similar image segmentation model and propose a cellular coevolutionary algorithm to optimize the model in a distributed way. Methods based on ant colonies and artificial life algorithms are also investigated for image segmentation and clustering problems [7].

The application of heuristic methods on image segmentation looks very promising, since segmentation can be seen as a clustering and combinatorial problem. Throughout this paper, we will consider the clustering problem and the segmentation problem as being similar. Accordingly, we consider solution methods for both problems interchangeably.

In this paper, a system based on an evolutionary algorithm, which is reminiscent of the well-know 'Life' game, invented by John Horton Conway [8], is presented. The evolution involves the colonization of a bidimensional world by a number of populations, which represent the different regions which are present in the image. The individuals, belonging to different populations, compete to occupy all the available space and adapt to the local environmental characteristics of the world. The details of the algorithm, introduced in a previous work [9] are reported in Section 2.

In this work we focus on the identification of a set of parameters which is suitable to solve a given task, pointing out the relationships between the parameters and the behavior of the evolving population. In order to better evaluate the performances of the algorithm, the identification phase has been carried out, in a first stage, on a set of synthetic, although realistic, images (Section 3). This allows to know the desired segmentation results, and to give a quantitative evaluation, through the analysis of confusion matrices. In a second phase, (Section 4) the method has been applied to a set of food images.

2 System Architecture

The system is based on an evolutionary algorithm which simulates the colonization of a bidimensional world by a number of populations. The world is organized in a bidimensional array of locations, or cells, where each cell is always occupied by an individual.

The world is represented by a matrix, associated with a vector of input images I_z (i.e. RGB components, textural parameters, or whatever), which are stacked one above the other. Each cell of the matrix corresponds to a pixel of the image stack, and therefore, the cell having coordinates $P = (x, y)$ is associated to a vector of features $e(x, y) = \{I_z(x, y)\}$. In our simulation, this feature vector is assumed to represent the environmental conditions at point P of our world.

During each generation, each individual has a variable probability S_r, depending both on the environmental conditions and on the local neighborhood, to survive to the next generation. When the individual fails to survive, the empty cell is immediately occupied by a newly generated individual.

2.1 Environmental Constraints

The environmental conditions in a cell influence the probability of the individual surviving in that location. If the population (which the individual belongs to) is well suited to the proposed environment, the survival chances of that individual are very high. On the other hand, if the population is suited to an environment which is very different from the local one, the possibilities for that individual to survive to the next generation are very low.

This requires us to define an ideal environment which maximizes the chances of survival of an individual of a given population. This ideal environment has been obtained by averaging, in each iteration, the environment in all the cells occupied by individual of the population.

For instance, if the population A is composed of individuals mainly located in dark zones of the input image, the few individuals belonging to the population A, which are situated in bright zones of the input image have a low survival rate. After a few iterations, the percentage of individual situated in dark areas will be increased.

A second parameter used to increase the selective pressure over the population is the variance of the feature vector. This is used to normalize the evaluation of the similarity between the ideal environment and the local environmental conditions.

The environmental factor described above has been modeled in our system by means of a survival factor S_e which is represented, for an individual belonging to the population i and situated in the point (x, y), as:

$$S_e = \frac{1}{1 + \exp \frac{c_i(x,y) - c_t}{c_0}} + m_e \tag{1}$$

The expression above represents a sigmoid-like function, centered in c_t. Parameters c_t and c_0 describe the position and the steepness of the sigmoid function, while the constant m_e represents a minimal survival rate. The variable $c_i(x, y)$ represents the similarity between the local environment $e(x, y)$ and the ideal environment e_i for the population i, evaluated as:

$$c_i(x,y) = \left| \frac{e(x,y) - e_i}{\sigma_i} \right| \tag{2}$$

where, as described above, e_i and σ_i are, respectively, the mean and the standard deviation of $e(x, y)$ over all points of the image occupied by individuals belonging to the population i.

2.2 Neighborhood Constraints

The presence of individuals of the same population in a neighborhood is known to increase the survival rate of them. In our simulation, this has been taken into account by including in the model a survival factor S_n which depends on the number of individuals n_i in a 3×3 neighborhood which belong to the same population of the individual located in the position (x, y).

The neighbor factor associated, named S_n, is evaluated as:

$$S_n = \frac{1}{1 + \exp \frac{n_t - n_i(x,y)}{n_0}} + m_s \tag{3}$$

where, as above, parameters n_t and n_0 describe the position and the steepness of the sigmoid function, while the constant m_s represents a minimal survival rate. It is worth noting the difference between the two survival rates is in the sign: in this case the survival rate increases when n_i increases, while S_e decreases when c_i increases.

2.3 Splitting and Merging

As presented above, the method does not prevent the situation were two populations are competing to colonize regions having similar environmental constraints. We overcome this problem by including, once over a predefined number of iterations, a split and merge step. In this step, we evaluate how different the separation are from each other by means of a statistical analysis of the populations descriptors. For each pair (i, j) of populations we evaluate a separation coefficient s_f as:

$$s_f = \sum_z \frac{(e_i - e_j)^2}{\sigma_i \sigma_j} \tag{4}$$

when this coefficient is too small, we assume that the two populations are statistically equivalent, and we merge them in a single one. At the same time, in order to preserve the total number of populations, the population having the highest dishomogeneity, measured as the largest value of $|\sigma_i|$, is split in two new populations.

2.4 Algorithm

The algorithm can be described according to the following steps:

1. On each point in the image is placed a random individual
2. For each generation:
   a. The average feature vector e_i and its standard deviation σ_i are computed for each population
   b. For each individual:
      i. The survival probability is computed as $S_r = S_e * S_n$.
      ii. If the individual does not survive, a new one replaces it. The new individual is assigned to a population randomly selected with probabilities proportional to the survival factor S_e of an individual of each population.
3. The separation s_f among populations is evaluated, and split and merge operation are performed

2.5 Evaluation Method

The evaluation of the segmentation results has been carried out by means of a segmentation cost C. The definition of the segmentation cost has been done, accordingly to the method proposed by Andrey [4], by introducing an over-segmentation cost and an under-segmentation one. These two components take in account the two properties that a segmentation algorithm must satisfy: (a) each region is homogeneous; and (b) neighboring regions cannot be merged into a larger homogeneous one.

The over-segmentation cost C^+ is an average measure of the extent to which a region in the target segmentation overlaps with more than one region of the candidate segmentation, a pattern that violates condition (b) above. Consider a region r in the target segmentation and a region l of the candidate segmentation and let $p_r(l)$ be the proportion of sites belonging to r that are assigned to l. If the candidate segmentation is identical to the target, then all frequencies $p_r(l)$ but one are equal to zero. By contrast, if r is segmented into a number of equally sized regions, then all $p_r(l)$ are equal. Hence, the entropy of distribution p_r is a measure of over-segmentation in region r. The total over-segmentation cost is defined as a weighted sum of individual over-segmentations. The contribution of each region r is weighted by the proportion α_r of sites belonging to r:

$$C^+ = -\sum_r \alpha_r \sum_l p_r(l) \log p_r(l) \tag{5}$$

The under-segmentation cost C^- is an average measure of the extent to which a region in the candidate segmentation overlaps with several regions of the target segmentation, a pattern that violates condition (a) above. Among the sites that are assigned to a region l in the candidate segmentation, we let $q_l(r)$ denote the proportion of sites that belong to region r in the target segmentation. Ideally, all $q_l(r)$ but one are equal to zero. By contrast, if l overlaps equally with all regions of the target, then all $q_l(r)$ are equal. Therefore, the entropy of the distribution q_l is a measure of the contribution of l to under-segmentation. The total under-segmentation cost C^- is defined as a weighted sum of individual under-segmentations. The contribution of each candidate region l is weighted by the proportion β_l of sites that have been attributed to l:

$$C^- = -\sum_l \beta_l \sum_r q_l(r) \log q_l(r) \tag{6}$$

For a candidate segmentation, we finally define a global segmentation cost as the summed under- and over-segmentation costs. The best segmentation cost is obviously zero.

3 Experiments with Simulated Images

This experiment uses a simulated magnetic resonance (MR) brain image obtained from the BrainWeb Simulated Brain Database [10]. The brain image was

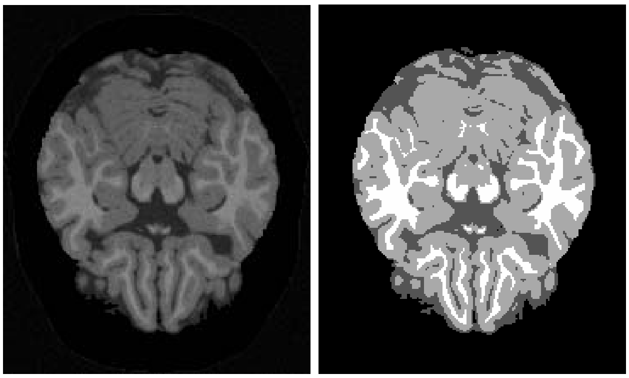


Fig. 1. Example of simulated MR brain images. Left: original image. Right: ground truth.

simulated with T1-weighted contrast, $1mm$ cubic voxels, 3% noise and no intensity inhomogeneity. The simulation is based on an anatomical model of normal brain, which can serve as the ground truth for any analysis procedure. The volume contains $181 \times 217 \times 181$ voxels and covers the brain completely. A transverse slice has been extracted from the volume. The non-brain parts of the image such as bone, cortex and fat tissue has been firstly removed. For our experiments we considered four classes, corresponding to Grey Matter (GM), White Matter (WM), CerebroSpinal Fluid (CSF) and background (BG).

The behavior of our algorithm can be evaluated with respect to all parameters, by comparing the output against the ground truth provided by the BrainWeb Simulated Brain Database [10].

Figure 1 shows a slice from the simulated data set and the relative ground truth.

3.1 Parameter Selection

The proposed algorithm depends on several parameters which describe the behavior of the survival rates in dependence of environmental and neighboring constraints. It is therefore of great importance to analyze how the performances of the method changes with respect to a variation in any of the parameters.

Although a complete analysis requires to explore the whole parameter space, in this work we describe a set of experiments with deals with the most important parameters one at a time. Each set of experiments describes the variation of the segmentation cost in dependence of one of parameters. To improve the statistical significancy of the test, the cost function has been averaged over a set of 10 runs for each value of the parameter.

The first group of experiments have been aimed to determine the importance of environmental parameters in the evolution of the populations.

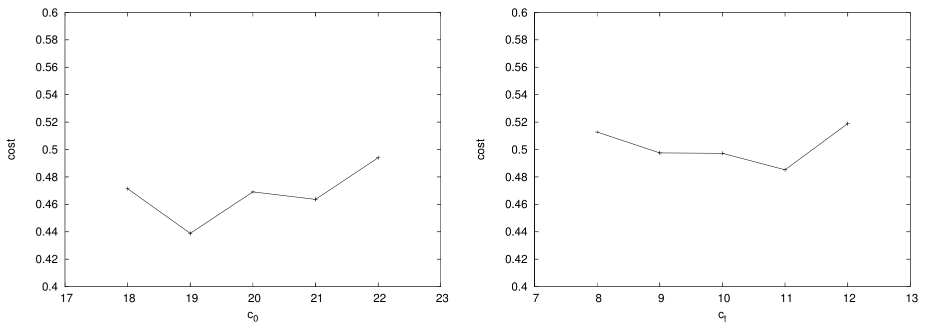


Fig. 2. Average segmentation cost as a function of environmental influence, described through the parameters c_0 (left) and c_t (right)

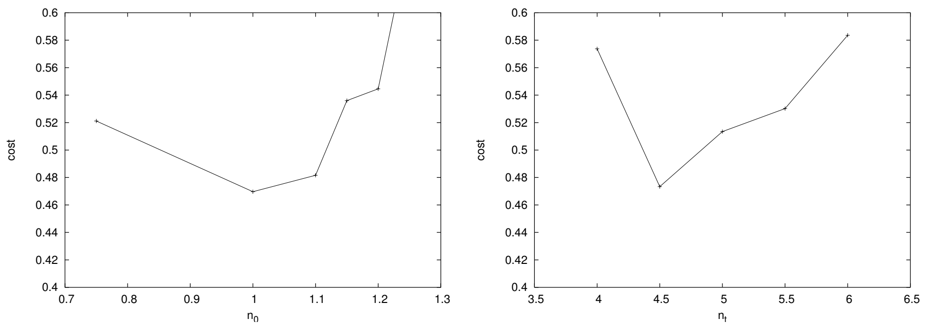


Fig. 3. Average segmentation cost as a function of neighborhood influence, described through the parameters n_0 (left) and n_t (right)

The value of c_t being fixed ($c_t = 10$), the average cost function has been determined for $c_0 \in [18, 22]$. Reciprocally, the value of c_0 being fixed ($c_0 = 20$), the average cost function has been determined for $c_t \in [8, 12]$.

Figure 2, on the left, describes the behavior of the average cost function with the environmental parameter c_0. Accordingly to (1), parameter c_t represents the steepness of the sigmoid function which describes S_e, where smaller values of the parameter correspond to an steepest curve. The plot shows that the optimal value is located approximately at $c_0 = 19$. The plot on the right reports the analysis of the behavior of the segmentation cost when the parameter c_t is varied. Accordingly to (1), the parameter c_t represents the value of the weighted difference c_i corresponding to a survival rate S_e equal to 0.5. The plot shows that the optimal value of c_t for the given problem is $c_t = 11$

In the second set of experiments, the value of n_t being fixed ($n_t = 4.5$), the average cost function has been determined for $n_0 \in [0.75, 1.25]$. Reciprocally, the value of n_0 being fixed ($n_0 = 1$), the average cost function has been determined for $n_t \in [4, 6]$.

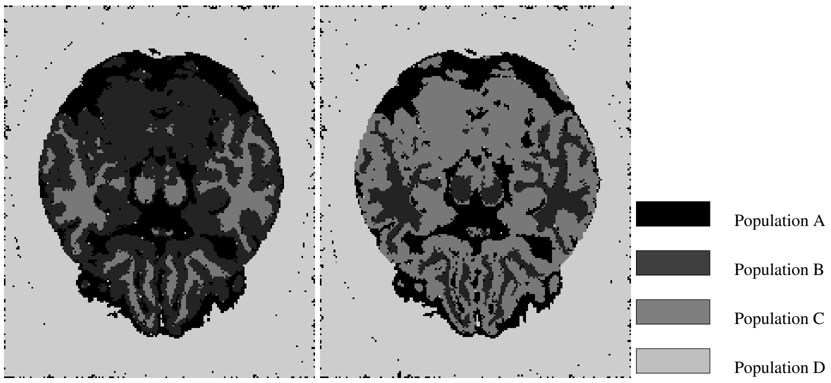


Fig. 4. Segmentation results on simulated MR brain images

Table 1. Confusion matrices corresponding to images in Figure 4

Actual	Classified as			
class	A	B	C	D
A	98	1	0	0
B	2	94	6	3
C	0	4	89	4
D	0	1	5	93

Actual	Classified as			
class	A	B	C	D
A	98	1	0	0
B	2	93	7	4
C	0	5	88	4
D	0	1	5	92

Figure 3 shows the variation of the segmentation cost with respect to the parameters describing neighborhood influence, namely n_0 and n_t. Both plots show how each parameter has an optimal value, which corresponds to a minimum of the segmentation cost. Comparing Figure 3 with Figure 2, the strongest influence of the neighborhood constraints over the segmentation cost can be noted, as indicated by the presence of a deeper minimum in the plots in Figure 3.

This result suggests the high importance for the colonization strategy of the presence of a neighborhood of individuals of the same type. When the value of S_n is too low, the individuals tend to group together irrespective of the actual environment of the image, as the only chance to survive is to form a compact region of individual of the same type, in order to increase the value of S_n. In the same way, when the values of S_n are too high, the grouping pressure diminishes, and the segmentation strategy becomes highly sensitive to noise.

3.2 Experimental Results

Our algorithm has been applied to segment these simulated MR brain images in four classes: GM, WM, CSF and BG. The number of populations is fixed to 4 and the algorithm is stopped after 200 generations.

Some examples of the segmentation results obtained with the optimal parameter set determined in this study ($n_t = 4.5$, $n_0 = 1$, $c_t = 11$, $c_0 = 19$, $\min(s_f) = 1$, $m_e = 0.1$, $m_s = 0.5$) are shown in Figure 4. Both images show that the obtained segmentation is in strong agreement with the ground truth image, reported in Figure 1, as indicated also from the confusion matrices, which are reported in Table 1.

4 Application to Food Images

Intramuscular fat content in meat influences some important meat quality parameters. For example, the quantitative intramuscular fat content has been shown to influence the palatability characteristics of meat. In addition, the visual appearance of the fat does influence the consumers overall acceptability of meat and therefore the choice when selecting meat before buying. Therefore the aim of the present application was to quantify intramuscular fat content in beef together with the visual appearance of fat in meat, and to compare the fat percentage measured by image analysis with chemical and sensory properties. Moreover the distribution of fat is an important criterion for meat quality evaluation and its expected palatability. Segmentation of meat images is the first step of this study.

The algorithm described in the previous section has been applied to meat image in order to obtain a proper classification and perform subsequent analysis.

Color images of M. longissimus dorsi were captured by a Sony DCS-D700 camera. The same exposure and focal distance were used for all images. Digital color photography was carried out with a Hama repro equipment (Germany). Green color was used as background and photographs were taken on both sides of the meat. The meat pieces were enlighten with two lamps, with two fluorescent tubes each (15 W). Polaroid filters were used on the lamps and on the camera.

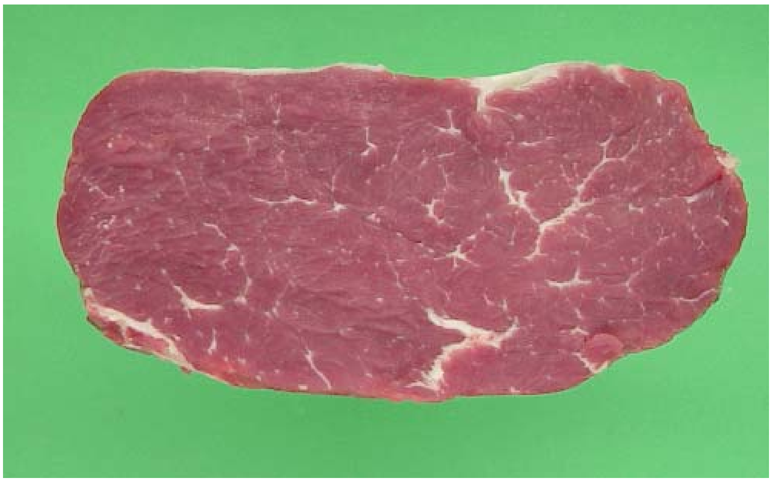


Fig. 5. Digital camera image of the *longissimus dorsi* muscle from representative beef meat (original is in color)

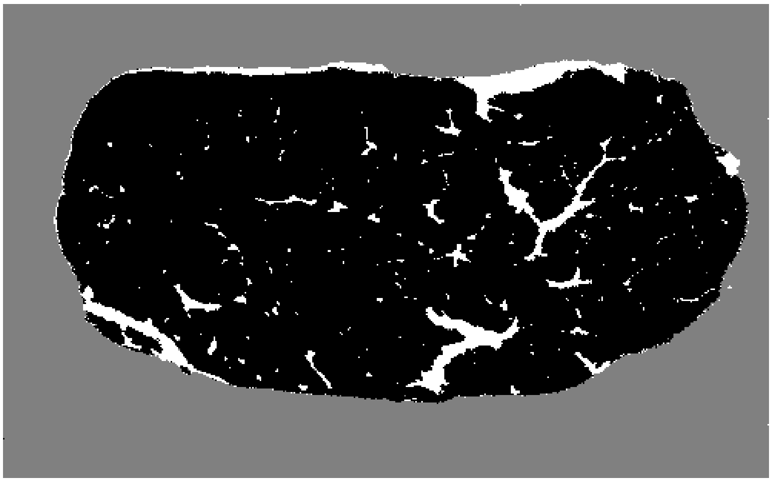


Fig. 6. Segmentation of the meat image shown in Figure 5

Images were 1344 x 1024 pixel matrices with a resolution of 0.13 x 0.13 mm (see Figure 5, as an example).

Our algorithm has been applied to segment these images in three classes: fat, muscle and background. The following parameters has been used: $n_t = 5$, $n_0 = 1$, $c_t = 10$, $c_0 = 20$, $\min(s_f) = 1$, $m_e = 0.1$, $m_s = 0.5$.

Figure 6 shows the results obtained after 200 iterations.

Unfortunately this algorithm is not able to distinguish between fat and connective tissue, as they have exactly the same color. A combination of the present algorithm with a previous algorithm [11] could provide good results also as concern the separation between fat and connective tissue. The percentage of fat extracted by the method proposed in this work was compared to the percentage measured by chemical analysis. We observed that advanced image analysis is useful for approximate measures of intramuscular fat content, even if the percentage of fat is usually overestimated, probably due to that digital photographs only reflect the meat surface.

5 Conclusions

In this paper, we presented an evolutionary algorithm for image segmentation. A segmentation cost was used to evaluate results and determine the optimal parameter combination. The proposed algorithm can be used for the segmentation of gray-scale, color and textured images. In particular segmentation of textured images can be obtained either by using a feature vector computed from statistical properties of texture, either combining our method with Markov Random Fields.

The representation of the environmental constraints as a feature vector allows us to easily extend the method to any vector-valued parametric images, independently on the number of components. Moreover, the normalization of each component of the similarity term c_i enables to use parametric images having different ranges of values.

In comparison with other contributions, it is worth to note that our evolutionary method does not include crossover and mutation operators, unlike Andrey's approach [4]. In common, we have that the population represents a candidate segmentation, and both methods are generic and can be applied to segmentation according any criterion.

We are planning to extend the method in order to include local properties on each population in the evaluation of survival rates. In this way we will enable the system to better adapt to slow variations present in the image (for instance, uneven illumination) which cannot be captured by the overall mean on the population. This extension could also allow for the introduction of a new split procedure based on the differentiation between local properties and the overall mean. In this way, it could be possible to automatically determine the number of segmentation classes, that now is decided by the user.

As concern the application to meat images, despite the fact that it is very difficult to achieve exact measures, probably due to digital photographs only reflect the meat surface, we believe that machine vision technology can provide an important tool for the food industry by allowing quantification of the visual appearance of meat, such as number of and size distribution of fat regions that are impossible to measure by chemical analysis. We also believe, on the basis of the obtained results, that the combined use of measurements of fat percentage and distribution can lead to an accurate description of meat quality, which will be investigated during the next phase of this study.

Acknowledgments

We would like to thank the meat-group at the Dept. of Food Science, Swedish University of Agricultural Sciences, Uppsala, especially Professor Kerstin Lundström and Maria Lundesjö Ahnström for providing the camera pictures and for their friendly collaboration.

References

1. Pal, N.R., Pal, S.K.: A review on image segmentation techniques. Pattern Recognition **26** (1993) 1277–1294
2. Bhanu, B., Lee, S., Ming, J.: Adaptive image segmentation using a genetic algorithm. IEEE Transactions on Systems, Man and Cybernetics **25** (1995) 1543–1567
3. Bhandarkar, S.M., Zhang, H.: Image segmentation using evolutionary computation. IEEE Transactions on Evolutionary Computation **3** (1999) 1–21
4. Andrey, P.: Selectionist relaxation: Genetic algorithms applied to image segmentation. Image and Vision Computing **17** (1999) 175–187
5. Liu, J., Tang, Y.Y.: Adaptive image segmentation with distributed behavior-based agents. IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (1999) 544–551

6. Veenman, C.J., Reinders, M.J.T., Backer, E.: A cellular coevolutionary algorithm for image segmentation. IEEE Transactions on Image Processing **12** (2003) 304–313
7. Ramos, V., Almeida, F.: Artificial ant colonies in digital image habitats - a mass behaviour effect study on pattern recognition. In: Proc. of ANTS'2000 - 2nd Int. Workshop on Ant Algorithms (From Ant Colonies to Artificial Ants), Brussels, Belgium (2000) 113–116
8. Gardner, M.: The fantastic combinations of John Conway's new solitaire game "life". Scientifican American **223** (1970) 120–123
9. Bocchi, L., Ballerini, L., Hässler: A new evolutionary algorithm for image segmentation. In: Application of Evolutionary Computation. Number 3449 in Lectures Notes in Computer Science, Lausanne, Switzerland (2005) 264–273
10. Collins, D.L., Zijdenbos, A.P., Kollokian, V., Sled, J.G., Kabani, N.J., Holmes, C.J., Evans, A.C.: Design and construction of a realistic digital brain phantom. IEEE Transactions on Medical Imaging **17** (1998) 463–468
11. Ballerini, L.: Genetic snakes for color images segmentation. In: Application of Evolutionary Computation. Volume 2037 of Lectures Notes in Computer Science., Milan, Italy (2001) 268–277

Enhancement of an Automatic Fingerprint Identification System Using a Genetic Algorithm and Genetic Programming

Wannasak Wetcharaporn[1], Nachol Chaiyaratana[1], and Sanpachai Huvanandana[2]

[1] Research and Development Center for Intelligent Systems, King Mongkut's Institute of Technology North Bangkok, 1518 Piboolsongkram Road, Bangsue, Bangkok 10800, Thailand
w_wannasak@hotmail.com, nchl@kmitnb.ac.th

[2] Department of Electrical Engineering, Chulachomklao Royal Military Academy, Suwanasorn Road, Muang, Nakhonnayok 26001, Thailand
shuvanan@crma.ac.th

Abstract. This paper presents the use of a genetic algorithm and genetic programming for the enhancement of an automatic fingerprint identification system (AFIS). The recognition engine within the original system functions by transforming the input fingerprint into a feature vector or fingercode using a Gabor filter bank and attempting to create the best match between the input fingercode and the database fingercodes. A decision to either accept or reject the input fingerprint is then carried out based upon whether the norm of the difference between the input fingercode and the best-matching database fingercode is within the threshold or not. The efficacy of the system is in general determined from the combined true acceptance and true rejection rates. In this investigation, a genetic algorithm is applied during the pruning of the fingercode while the search by genetic programming is executed for the purpose of creating a mathematical function that can be used as an alternative to the norm operator. The results indicate that with the use of both genetic algorithm and genetic programming the system performance has improved significantly.

1 Introduction

Biometrics is an automated technique for identifying individuals based upon their physical or behavioural characteristics. The physical characteristics that are generally utilised as biometrics cover faces, retinae, irises, fingerprints and hand geometry while the behavioural characteristics that can be used include handwritten signatures and voiceprints. Among various biometrics, fingerprint-based identification is the most mature and proven technique. A fingerprint is made up from patterns of ridges and furrows on the surface of a finger [1]. The uniqueness of a fingerprint can be explained via (a) the overall pattern of ridges and furrows and (b) the local ridge anomalies called minutiae points such as a ridge bifurcation and a ridge ending. As fingerprint sensors are nowadays getting smaller and cheaper, automatic fingerprint identification systems (AFISs) have become popular alternatives or complements to traditional identification methods. Examples of applications that have adopted an AFIS are ranging from security control with a relatively small database to criminal identification with a large database.

Research in the area of fingerprint-based identification can be divided into two categories: fingerprint classification and fingerprint recognition. The purpose of classification is to cluster a database of fingerprints into sub-categories where the sub-categories are in general defined according to a Henry system [2]. Several techniques including syntactic approaches [3, 4], structural approaches [5, 6, 7, 8, 9], neural network approaches [4, 10, 11, 12] and statistical approaches [13] have been successfully used in fingerprint classification. In contrast, the purpose of recognition is to match the fingerprint of interest to the identity of an individual. A fingerprint recognition system is widely used in security-related applications including personnel identification and access control. For the purpose of access control, the goal of recognition is (a) to identify correctly a system user from the input fingerprint and grant him or her an appropriate access and (b) to reject non-users or intruders. Various techniques including conventional minutiae-based approaches [14, 15, 16], evolutionary minutiae-based approaches [17, 18, 19] and texture-based approaches [20] have been applied to fingerprint recognition. Among these techniques, the approach involving the transformation of a fingerprint into a fingercode [20] has received much attention in recent years. In brief, a fingerprint is transformed via a Gabor filter-based algorithm where the resulting feature vector or fingercode is a fixed length string that is capable of capturing both local and global details in a fingerprint. The fingerprint recognition is then achieved by matching the fingercode interested with that in the database via a vector distance measurement. Since the fingerprint is now represented by a unique fixed length vector and the matching mechanism is carried out through a vector operation, this approach has proven to be reliable, fast and requiring a small database storage.

Although a number of impressive results have been reported in Jain et al. [20], the recognition capability of the fingercode system can be further enhanced. One possible approach to improve the system is to modify the fingercode using a feature pruning technique. In most pattern recognition applications, the original feature vector is often found to be containing a number of redundant features. Once these features are removed, the recognition efficacy is in general maintained or improved in some cases. The most direct advantage for pruning the fingercode is the reduction in the database storage requirement. The candidate technique for pruning the fingercode is a genetic algorithm [21] where the decision variables indicate the presence and absence of features while the optimisation objective is the recognition efficacy. In addition to the feature pruning approach, the recognition system can also be improved by modifying the fingercode matching mechanism. In the original work by Jain et al. [20], a vector distance between the input fingercode and the database fingercode is used to provide the degree of matching. As a result, the distance value from each feature will contribute equally to the judgment on how well two fingercodes match one another. In this investigation, the mathematical structure for obtaining the distance and the level of contribution from each feature will be manipulated and explored using a genetic programming technique [22]. This part of the investigation is carried out in order to further increase the recognition capability of the system from that achieved after the feature pruning.

The organisation of this paper is as follows. In section 2, a brief explanation on the original fingercode system will be given. This also includes the description of the fingercode, which is the feature vector, and the matching mechanism. The application of the genetic algorithm on the feature pruning and the results will be discussed in section 3. Following that, the use of the genetic programming in matching mechanism modification and the results will be given in sections 4 and 5. Finally, the conclusions are drawn in section 6.

2 Fingercode System

The fingercode system developed by Jain et al. [20] consists of two major stages: filter-based feature extraction and fingercode matching stages. These two components are explained as follows.

2.1 Filter-Based Feature Extraction

There are three main steps in the feature extraction process described in Jain et al. [20]: (a) determination of a reference frame from the fingerprint image, (b) filtering the image using a Gabor filter bank and (c) computation of the standard deviation of pixel values in sectors around the reference point in the filtered image to obtain a feature vector or fingercode. Firstly, the point of maximum curvature of ridges in the fingerprint image is initially located as the reference point. The reference axis is then defined as the axis of local symmetry at the reference point. Next, the region of interest is identified; the region is composed of n concentric bands around the reference point where each band is segmented into k sectors. In this paper, eight concentric bands are used and there are 16 sectors in each band. Thus there are a total of $16 \times 8 = 128$ sectors in the region of interest as shown in Fig. 1. The region of interest is filtered using a Gabor filter [23] where the standard deviation of filtered pixels within each sector is subsequently used as a feature in the fingercode. With the use of eight Gabor filters per one fingerprint image, the total number of features in a fingercode for this paper is $8 \times 128 = 1,024$.

2.2 Fingercode Matching

After a feature vector or fingercode has been extracted from the input fingerprint, an attempt to identify the best match between the fingercode obtained and that in the database is carried out. The best-matching fingercode from the database will be the one where a norm of the distance between itself and the

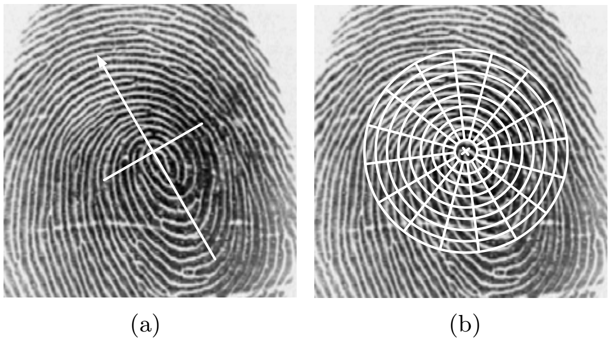


Fig. 1. (a) The reference axis (b) the reference point (×) and the region of interest, which consists of 128 sectors

input fingercode is minimal. In this paper, a 1-norm is used in the distance measurement. It is noted that other norms such as a Euclidean norm and an infinite-norm can also be used. The fingerprint recognition can then be carried out via comparing the best-matching norm with a threshold. If the norm is less than or equal to the threshold, the recognition will identify the input fingerprint as being a part of the database and hence belongs to one of the system users. On the other hand, if the norm exceeds the threshold, the system will reject the input fingerprint and decide that the fingerprint belongs to an intruder.

The efficacy of the fingercode system can be determined from the correctness of the system output after the matching procedure. False output from the system can generally be divided into two categories: a false acceptance and a false rejection. A false acceptance refers to the situation when the system identifies an input fingerprint as belonging to one of the users while in fact the fingerprint belongs to either another user or an intruder. In contrast, a false rejection refers to the case where the system falsely identifies an input fingerprint as belonging to an intruder while the fingerprint actually comes from one of the users. Hence, the system efficacy can be expressed in terms of the combined true acceptance and true rejection rates, a false acceptance rate (FAR) and a false rejection rate (FRR). In the following three sections, the improvement of the fingercode system by means of genetic algorithm and genetic programming searches will be given.

3 Feature Pruning Using a Genetic Algorithm

An attempt on reducing the number of features in a fingercode using a genetic algorithm is made. The decision variables for the optimisation cover the presence and absence of features in the fingercode. The decision variables can thus be represented by a binary chromosome where '1' represents the presence of a feature whilst '0' signifies the absence of a feature. In this investigation, there are 1,024 features in the fingercode. As a result, the chromosome length is also equal to 1,024 bits. Since the size of the reduced feature vector can vary during the optimisation process, the value of threshold required for the decision made by the recognition system has to also be modified accordingly. In this paper, a 1-norm is used during the feature matching procedure. The threshold can thus be set such that it is linearly proportional to the number of remaining features in the fingercode after pruning. After the matching between all input fingercodes and the database fingercodes, and the acceptance/rejection decision has been made, the fitness value of each chromosome can be calculated from the combined true acceptance and true rejection rates expressed in percent. In this investigation, 400 fingerprint images are collected from 40 individuals where each individual contributes ten fingerprints. During a genetic algorithm run, 300 fingerprints from 30 individuals are retained within the user database while the other 100 fingerprints from the remaining ten individuals are used as fingerprints from intruders. All 400 fingerprints are transformed into fingercodes using the feature extraction procedure where the decisions to accept or reject the input fingercodes are subsequently made. It is noted that the fingerprint database remains unchanged throughout the genetic algorithm run, which in this case is repeated ten times with different initial populations. The parameter setting for the genetic algorithm is summarised in Table 1. After all ten algorithm runs are completed, reduced fingercodes as represented by the best individual among all runs, and three additional chromosomes, which are resulted from applying an AND function, an OR function and a majority vote rule to aligned bits of all ten best individuals from different runs are then tested or validated. The fingerprint databases used for validation also comprise of fingerprints taken from the original 400 fingerprints. However, all except one fingerprint sets for validation would be different from that used during the genetic algorithm runs. The original database and nine newly created databases, which are obtained by swapping fingerprints between the original user and intruder databases according to the scheme displayed in Fig. 2, are utilised during the validation where the results are illustrated in Tables 2 and 3.

Table 1. Parameter setting for the genetic algorithm

Parameter	Setting and Value
Chromosome representation	Binary chromosome
Chromosome length	1,024
Fitness scaling method	Linear scaling
Selection method	Stochastic universal sampling
Crossover method	Uniform crossover
Crossover probability	0.8
Mutation method	Bit-flip mutation
Mutation probability	0.1
Population size	100
Number of generations	1,000
Number of repeated runs	10

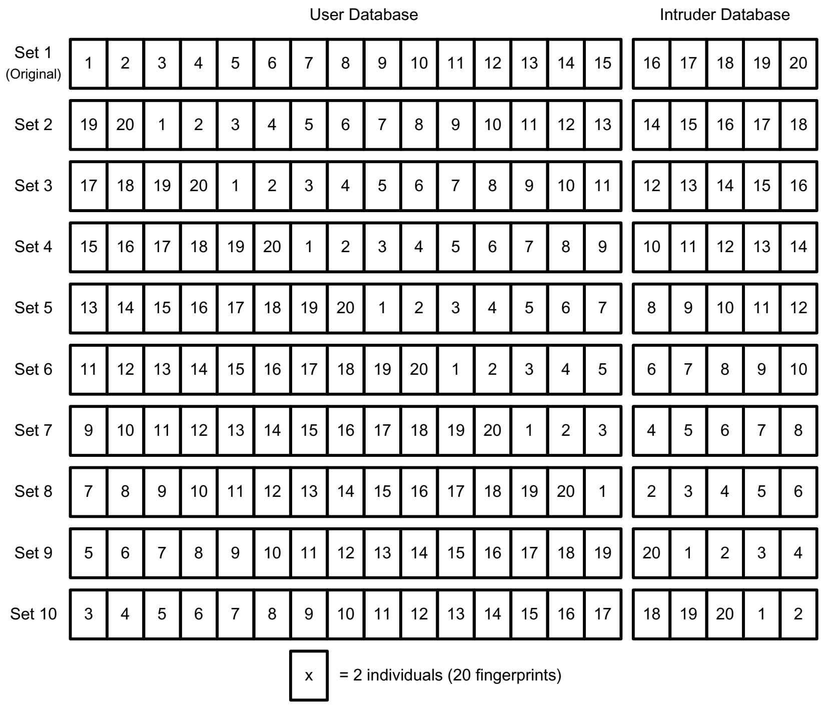

User Database
Intruder Database
Set 1 (Original) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | 16 17 18 19 20
Set 2 19 20 1 2 3 4 5 6 7 8 9 10 11 12 13 | 14 15 16 17 18
Set 3 17 18 19 20 1 2 3 4 5 6 7 8 9 10 11 | 12 13 14 15 16
Set 4 15 16 17 18 19 20 1 2 3 4 5 6 7 8 9 | 10 11 12 13 14
Set 5 13 14 15 16 17 18 19 20 1 2 3 4 5 6 7 | 8 9 10 11 12
Set 6 11 12 13 14 15 16 17 18 19 20 1 2 3 4 5 | 6 7 8 9 10
Set 7 9 10 11 12 13 14 15 16 17 18 19 20 1 2 3 | 4 5 6 7 8
Set 8 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1 | 2 3 4 5 6
Set 9 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 | 20 1 2 3 4
Set 10 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | 18 19 20 1 2
x = 2 individuals (20 fingerprints)

Fig. 2. Ten validation sets for testing the fingercode system

Table 2. Validation results of the fingercode system with and without feature pruning. The first validation set is also used during all repeated runs of the genetic algorithm.

	Recognition Efficacy (%)				
Validation Set	Original Fingercode (1,024 Features)	Evolutionary Fingercode (419 Features)	Fingercode from an AND Function (384 Features)	Fingercode from an OR Function (521 Features)	Fingercode from a Majority Vote (492 Features)
1	88.75	96.00	95.50	95.00	96.25
2	85.25	95.00	94.50	94.25	95.50
3	83.25	91.50	91.25	91.00	92.00
4	83.00	91.00	90.50	90.50	91.00
5	83.25	91.25	91.25	90.25	92.50
6	82.25	90.50	90.25	89.00	91.25
7	76.50	85.00	84.75	83.50	85.75
8	76.75	87.00	86.75	85.50	88.00
9	79.50	87.75	87.75	86.25	88.75
10	85.75	92.00	92.00	90.25	92.50
Average	82.43	90.70	90.45	89.55	91.35

Table 3. False acceptance and false rejection rates of the fingercode system with and without feature pruning

Validation Set	False Acceptance and False Rejection Rates (%)									
	Original Fingercode (1,024 Features)		Evolutionary Fingercode (419 Features)		Fingercode from an AND Function (384 Features)		Fingercode from an OR Function (521 Features)		Fingercode from a Majority Vote (492 Features)	
	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
1	1.50	9.75	0.75	3.25	1.00	3.50	0.75	4.25	0.75	3.00
2	1.50	13.25	0.75	4.25	1.00	4.50	0.75	5.00	0.75	3.75
3	0.25	16.50	0.00	8.50	0.25	8.50	0.00	9.00	0.25	7.75
4	0.25	16.75	0.00	9.00	0.75	8.75	0.00	9.50	1.00	8.00
5	0.25	16.50	0.00	8.75	0.00	8.75	0.00	9.75	0.00	7.50
6	0.25	17.50	0.00	9.50	0.25	9.50	0.00	11.00	0.50	8.25
7	4.25	19.25	5.00	10.00	5.25	10.00	5.00	11.50	5.50	8.75
8	2.25	21.00	2.50	10.50	2.75	10.50	2.50	12.00	2.75	9.25
9	2.25	18.25	2.50	9.75	2.50	9.75	2.50	11.25	2.50	8.75
10	1.25	13.00	2.25	5.75	2.25	5.75	2.50	7.25	2.50	5.00
Average	1.40	16.17	1.38	7.92	1.60	7.95	1.40	9.05	1.65	7.00

From Tables 2 and 3, it can be clearly seen that the use of a pruned or reduced fingercode leads to an improvement in recognition performance over the use of a full fingercode for at least 7% in overall. The highest improvement comes from the case of reduced fingercode obtained after using a majority vote rule where detailed results indicate that there is a significant improvement in the false rejection rate. On the other hand, the reduced fingercodes that have the worst performance are the ones resulted from the use of AND and OR functions. These results can be interpreted as follows. With the application of a majority vote rule in deciding whether a feature should be maintained or removed from the fingercode, the effect of uncertainties due to the stochastic search nature of genetic algorithms on the overall optimisation result would be minimised. During each genetic algorithm run, the search is conducted in a manner that maximises the recognition efficacy. Since the search is a stochastic one and there may be more than one globally optimal reduced fingercode, the use of a majority vote rule would help maintaining necessary features detected in most or all runs while at the same time eliminating possible redundant features. This reason is supported by the results where AND and OR functions are used, which indicate that there is no significant gain in recognition performance over the use of a reduced fingercode obtained from typical genetic algorithm runs. With the application of an OR function, the resulting fingercode would contain both necessary features and some redundant features while with the use of an AND function, some crucial features may be left out since they are not present in all best individuals. These two phenomena would have caused a reduction in the recognition performance.

4 Modification of the Matching Mechanism Using Genetic Programming

In the original work by Jain et al. [20] and the investigation so far, the decision to accept or reject input fingercode is based on whether the best-matching norm is within the threshold or not. In this section, the calculation of 1-norm will be replaced by the mathematical function or operation evolved by genetic programming (GP). Nonetheless, the output from the evolved function will still be compared with the threshold during the decision-making procedure. Since the use of a reduced fingercode generated by a majority vote rule has proven to produce the current best result, all features from this reduced fingercode will be used as a part of terminal set. The terminal set is thus made up from preset constant values and the absolute differences between the input features and the corresponding features from a reduced fingercode in the database. It is noted that the best-matching fingercode in the database is the one that the GP-evolved function returns the minimum value. The parameter setting for the genetic programming is summarised in Table 4.

Table 4. Parameter setting for the genetic programming

Parameter	Setting and Value
Tree initialisation method	Grow method
Maximum tree depth	10
Terminal set	{Constants: 0.25, 0.50, 0.75, 1.25, 1.50, 1.75, 2.00 and absolute differences between input and database features}
Function set	$\{+, -\}$
Fitness scaling method	Linear scaling
Selection method	Stochastic universal sampling
Crossover probability	0.8
Mutation probability	0.1
Population size	100
Number of elitist individuals	1
Number of generations	2,000
Number of repeated runs	10

Similar to the approach presented in the previous section, ten databases are also used during the validation where the results are displayed in Table 5. The genetic programming results are produced using the best individual among all ten runs. From Table 5, it can be clearly seen that the replacement of 1-norm by the GP-generated function leads to a further improvement in terms of the recognition efficacy, false acceptance rate and false rejection rate from that achieved earlier. This also implies that the use of mathematical functions other than a norm function may be more suitable to the fingercode system. It is noticeable that the system performance is highest in the case of the first validation set,

Table 5. Validation results of the reduced-feature fingercode system with the use of 1-norm and GP-generated function during matching

Validation Set	Matching via 1-Norm			Matching via GP-Generated Function		
	Efficacy (%)	FAR (%)	FRR (%)	Efficacy (%)	FAR (%)	FRR (%)
1	96.25	0.75	3.00	98.00	0.50	1.50
2	95.50	0.75	3.75	97.00	0.50	2.50
3	92.00	0.25	7.75	93.00	0.25	6.75
4	91.00	1.00	8.00	91.75	1.00	7.25
5	92.50	0.00	7.50	92.50	0.25	7.25
6	91.25	0.50	8.25	92.25	0.25	7.50
7	85.75	5.50	8.75	88.25	3.75	8.00
8	88.00	2.75	9.25	91.25	1.00	7.75
9	88.75	2.50	8.75	91.75	1.25	7.00
10	92.50	2.50	5.00	95.00	2.25	2.75
Average	91.35	1.65	7.00	93.08	1.10	5.82

where it is also the data set used during the evolution of a matching function using genetic programming.

5 Matching by a Combined 1-Norm and GP-Generated Operator

In the previous section, the GP-generated function is proven to outperform 1-norm during the recognition evaluation. However, the generated function is rather complex since the required tree depth is ten and the tree contains a large number of terminals including constants and 492 inputs from the reduced fingercode, which are the majority vote results of multiple genetic algorithm runs. Consequently, the tree evolution requires 2,000 generations, which takes reasonably large computational effort. One possible approach for reducing both tree size and computational burden is to create a hybrid matching mechanism that utilises both 1-norm and GP-generated function. From sub-section 2.1, the features contain in a fingercode are obtained after eight Gabor filters have been applied to a fingerprint image. The fingercode can thus be divided into eight code-groups where each code-group represents a part of the fingercode, which is obtained from the same Gabor filter. The features contained in a few code-groups can be used as parts of GP terminals while the features in the remaining code-groups are left as inputs to the 1-norm operator. The complete matching output would be the summation between the outputs from GP tree and 1-norm. Similar to the early approach, the complete output from the combined function will be compared with the threshold during the decision making process where the best-matching fingercode from the database is the one where the combined function gives the minimum value. In this part of the investigation, the parame-

Table 6. Validation results of the reduced-feature fingercode system with the use of GP-generated function and combined function during matching

Validation Set	Matching via GP-Generated Function			Matching via Combined Function		
	Efficacy (%)	FAR (%)	FRR (%)	Efficacy (%)	FAR (%)	FRR (%)
1	98.00	0.50	1.50	97.50	0.25	2.25
2	97.00	0.50	2.50	96.25	0.50	3.25
3	93.00	0.25	6.75	93.00	0.25	6.75
4	91.75	1.00	7.25	91.25	1.00	7.75
5	92.50	0.25	7.25	92.25	0.50	7.25
6	92.25	0.25	7.50	92.25	0.50	7.25
7	88.25	3.75	8.00	89.00	3.25	7.75
8	91.25	1.00	7.75	90.75	1.00	8.25
9	91.75	1.25	7.00	91.50	0.75	7.75
10	95.00	2.25	2.75	93.50	2.50	4.00
Average	93.08	1.10	5.82	92.73	1.05	6.23

ter setting for the genetic programming is similar to that given in Table 4 except that the maximum tree depth is reduced to seven and the number of generations is decreased to 300. Furthermore, the fingercode features, which are parts of GP terminals, come from only one code-group and are extracted from the reduced fingercode previously selected by a genetic algorithm together with a majority vote. The remaining features in the reduced fingercode are thus used as inputs to the 1-norm operator.

Ten previously described databases are utilised during the validation where the results are displayed in Table 6. The illustrated results are also produced using the best individual among all ten runs. From Table 6, it can be clearly seen that there is a slight drop in the recognition performance after replacing the original GP-generated function with the combined function. The main cause for this phenomenon is a slight increase in the false rejection rate in the results from most databases. Nonetheless, the results from the combined function are still better than that obtained using the 1-norm function in both false acceptance and false rejection rates. In addition, the GP tree is now less complex since both tree size and size of terminal set are significantly reduced.

6 Conclusions

In this paper, an improvement on an automatic fingerprint identification system (AFIS), which can be referred to as a fingercode system [20] is illustrated. The original system employed a recognition engine and can be used to identify system users and reject intruders. The fingercode, which is a fixed length feature vector, is extracted from a fingerprint using a Gabor filter-based algorithm. The decision to either reject or accept the input fingercode is subsequently made based on whether the norm of the difference between the input fingercode and the best-matching fingercode from the database exceeds the preset threshold or not. The efficacy of the fingercode is measured in terms of the combined true acceptance and true rejection rates. Two approaches for system enhancement have been proposed. In the first approach, the full feature vector or fingercode is pruned using a genetic algorithm. After the elimination of redundant features, the use of reduced fingercode is proven to help improving the system performance. In the second approach, the calculation of norm during the fingercode matching procedure is either partially or fully replaced by a mathematical function or operation evolved by genetic programming (GP). With the use of features from the reduced fingercode as parts of the terminal set, the GP-evolved function in the recognition engine is proven to be highly efficient. As a result, the recognition capability of the system is further increased from that achieved earlier by pruning the fingercode. This helps to prove that the use of both genetic algorithm and genetic programming can significantly improve the recognition performance of the fingercode system.

Acknowledgements

This work was supported by the Thailand Research Fund (TRF) through the Research Career Development Grant and the National Science and Technology Development Agency (NSTDA) through the Thailand Graduate Institute of Science and Technology (TGIST) programme.

References

1. Galton, F.: Finger Prints. Macmillan, London, UK (1892)
2. Henry, E.R.: Classification and Uses of Finger Prints. HM Stationary Office, London, UK (1905)
3. Moayer, B., Fu, K.S.: A tree system approach for fingerprint pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **8**(3) (1986) 376–387
4. Blue, J.L., Candela, G.T., Grother, P.J., Chellappa, R., Wilson, C.L.: Evaluation of pattern classifiers for fingerprint and OCR applications. Pattern Recognition **27**(4) (1994) 485–501
5. Rao, T.C.M.: Feature extraction for fingerprint classification. Pattern Recognition **8**(3) (1976) 181–192
6. Hrechak, A.K., McHugh, J.A.: Automated fingerprint recognition using structural matching. Pattern Recognition **23**(8) (1990) 893–904
7. Karu, K., Jain, A.K.: Fingerprint classification. Pattern Recognition **29**(3) (1996) 389–404
8. Hong, L., Jain, A.K.: Classification of fingerprint images. In: Proceedings of the 11th Scandinavian Conference on Image Analysis, Kangerlussuaq, Greenland (1999)
9. Cho, B.-H., Kim, J.-S., Bae, J.-H., Bae, I.-G., Yoo, K.-Y.: Core-based fingerprint image classification. In: Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain (2000) 859–862

10. Mitra, S., Pal, S.K., Kundu, M.K.: Fingerprint classification using a fuzzy multilayer perceptron. Neural Computing and Applications **2**(4) (1994) 227–233
11. Halici, U., Ongun, G.: Fingerprint classification through self-organizing feature maps modified to treat uncertainties. Proceedings of the IEEE **84**(10) (1996) 1497–1512
12. Jain, A.K., Prabhakar, S., Hong, L.: A multichannel approach to fingerprint classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **21**(4) (1999) 348–359
13. Coetzee, L., Botha, E.C.: Fingerprint recognition in low quality images. Pattern Recognition **26**(10) (1993) 1441–1460
14. Farina, A., Kovács-Vajna, Z.M., Leone, A.: Fingerprint minutiae extraction from skeletonized binary images. Pattern Recognition **32**(5) (1999) 877–889
15. Fan, K.C., Liu, C.W., Wang, Y.K.: A randomized approach with geometric constraints to fingerprint verification. Pattern Recognition **33**(11) (2000) 1793–1803
16. Tan, X., Bhanu, B.: Robust fingerprint identification. In: Proceedings of the 2002 International Conference on Image Processing, Rochester, NY (2002) I-277–I-280
17. Tan, X., Bhanu, B.: Fingerprint matching by genetic algorithms. In: Late Breaking Papers at the 2002 Genetic and Evolutionary Computation Conference, New York, NY (2002) 435–442
18. Tan, X., Bhanu, B.: Fingerprint verification using genetic algorithms. In: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision, Orlando, FL (2002) 79–83
19. Tan, X., Bhanu, B.: Fingerprint matching by genetic algorithms. Pattern Recognition **39**(3) (2006) 465–477
20. Jain, A.K., Prabhakar, S., Hong, L., Pankanti, S.: Filterbank-based fingerprint matching. IEEE Transactions on Image Processing **9**(5) (2000) 846–859
21. Goldberg, D.E.: Genetic Algorithms: In Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)
22. Koza, J.R.: Genetic Programming: On the Programming by Computers by Means of Natural Selection. MIT Press, Cambridge, MA (1992)
23. Daugman, J.G.: High confidence visual recognition of persons by a test of statistical independence. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(11) (1993) 1148–1161

Evolutionary Singularity Filter Bank Optimization for Fingerprint Image Enhancement

Ung-Keun Cho, Jin-Hyuk Hong, and Sung-Bae Cho

Dept. of Computer Science, Yonsei University,
Biometrics Engineering Research Center,
134 Sinchon-dong, Seodaemun-ku,
Seoul 120-749, Korea
{bearoot, hjinh}@sclab.yonsei.ac.kr,
sbcho@cs.yonsei.ac.kr

Abstract. Singularity is the special feature of fingerprints for identification and classification. Since the performance of singularity extraction depends on the quality of fingerprint images, image enhancement is required to improve the performance. Image enhancement with various image filters might be more useful than a filter, but it is very difficult to find a set of appropriate filters. In this paper, we propose a method that uses the genetic algorithm to find those filters for superior performance of singularity extraction. The performance of the proposed method has been verified by the experiment with NIST DB 4. Moreover, the proposed method does not need any expert knowledge to find the type and order of filters for the target domain, it can be easily applied to other applications of image processing.

1 Introduction

AFIS (Automatic Fingerprint Identification System) finds out whether an individual's incoming fingerprint image is the same as any of the target templates in the database. Classification, which groups fingerprints into predefined several categories based on the basis of global ridge pattern, can reduce the number of fingerprints for matching in the large database. Singular points are conceptualized as aid for fingerprint classification by Henry [1], and used for better identification [2]. The accuracy of singularity extraction is subject to the quality of images, so that it is hard to achieve good performance. There are lots of methods for singularity extraction but also the improvement of the quality [3].

Segmentation [4] and enhancement [5] to improve the quality of fingerprint images. Segmentation classifies a part of the image into a background or fingerprint region, so as to discard the background regions to reduce the number of false features extracted. Enhancement improves the clarity of the ridge structures to extract correct features. There are many filters for enhancement. For fingerprint images, enhancement using various filters together might be better than when using only one, but usually it requires the expert knowledge to determine the type and order of filters [5,6,7,8].

Since it is actually impossible to examine all possible combinations of filters, it needs a heuristic algorithm. In this paper, we exploit the genetic algorithm [9,10,11,12] to find out a set of proper filters to enhance the fingerprint images.

2 Related Work

2.1 Singularity

Fingerprints contain singularities that are known as core and delta points. The core is the topmost point on the innermost recurving ridge, while the delta is the center of a triangular region where three different direction flows meet. In the process of fingerprint classification, singularity is used directly [13], or with ridge patterns [14,15]. It was also used as landmarks for the other features [16,17]. There are other applications of singularity such as landmarks for fingerprint matching [18].

The Poincare index is a popular method to detect singularity using orientation field [3]. Fig. 1 shows the orientation field of core and delta regions. The orientation field is estimated for each block, and it is subject to the quality of the image. Image enhancement to improve the quality is required to calculate the orientation field correctly.

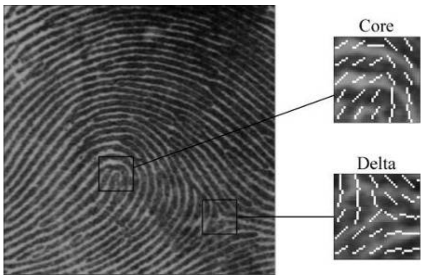


Fig. 1. The orientation field of core and delta regions

2.2 Image Filter

The goal of general image processing is to detect objects on image and to classify them through the analysis of images. In images, low quality is hard to correctly extract features. Filtering, such as reducing image noises, smoothing, removing some forms of misfocus and motion blur, is in the front step of image processing. Typically, there are histogram-based, mask-based and morphology-based image filters [19]. Many methods have used these filters to improve quality of fingerprint image. Since Hong [5] introduced the Gabor filter to enhance fingerprint images, it has been adopted in many methods for fingerprint enhancement [6,7,8]. Fig. 2 shows example images obtained by several filters.

2.3 Genetic Algorithm

The genetic algorithm is an evolutionary algorithm, which is based on mechanisms of natural selection and the survival of fittest. In each generation, a new set of individuals

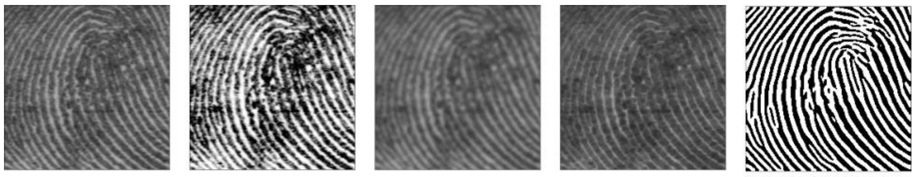


Fig. 2. Example images after filtering

is generated by selection, crossover, and mutation of previous ones. An individual represents a solution to the problem by a list of parameters, called chromosome or genome. The first population, a set of chromosomes, is initialized randomly, while population in the next generation is generated from population of the previous generation. The population is stochastically generated using genetic operators such as selection, crossover, and mutation with the fitness measures of current population. Generally the average fitness will have increased by this procedure for the population. The genetic algorithm has been successfully applied in many search, optimization, and machine learning problems [9,10].

3 Image Enhancement Based on Genetic Algorithm

For the correct feature extraction, the quality of the image should be improved by using appropriate image filters. The number of constructing an ordered subset of n filters from a set of m filters is given by m^n. Trying all cases to find out the best one practically impossible when there are lots of filters available. In this paper, a genetic algorithm is used to search filters of the proper type and order.

Fig. 3 shows the procedure of the proposed method which also presents the process of evaluating fitness. In each generation, the fitness of chromosome is evaluated by using the fitness function, and chromosomes with higher fitness are stochastically selected and applied with genetic operators such as crossover and mutation to reproduce the population of the next generation. Elitist-strategy [20] that always keeps the best chromosome found so far is used. Chromosomes are represented as simple numbers corresponding with individual filters, and Table 1 shows the type and effect of 71 individual filters.

Chromosomes with the length of five represent a set of filters. Fig. 4 shows the structure of chromosomes and the examples of genetic operators such as crossover and mutation.

The fitness of an individual is estimated by the performance of singularity extraction using the Poincare index. The Poincare index extracts singularity using the orientation field which is calculated for each block. Singularity is classified into the core and delta, and these two points do not lie together in the same location. Let $S_d = \{s_{d1}, s_{d2}, \ldots, s_{dn}\}$ be the set of n singularity points detected by the singularity extraction algorithm, and $S_e = \{s_{e1}, s_{e2}, \ldots, s_{em}\}$ be the set of m singularity points identified by human experts in an input fingerprint image. The following sets are defined.

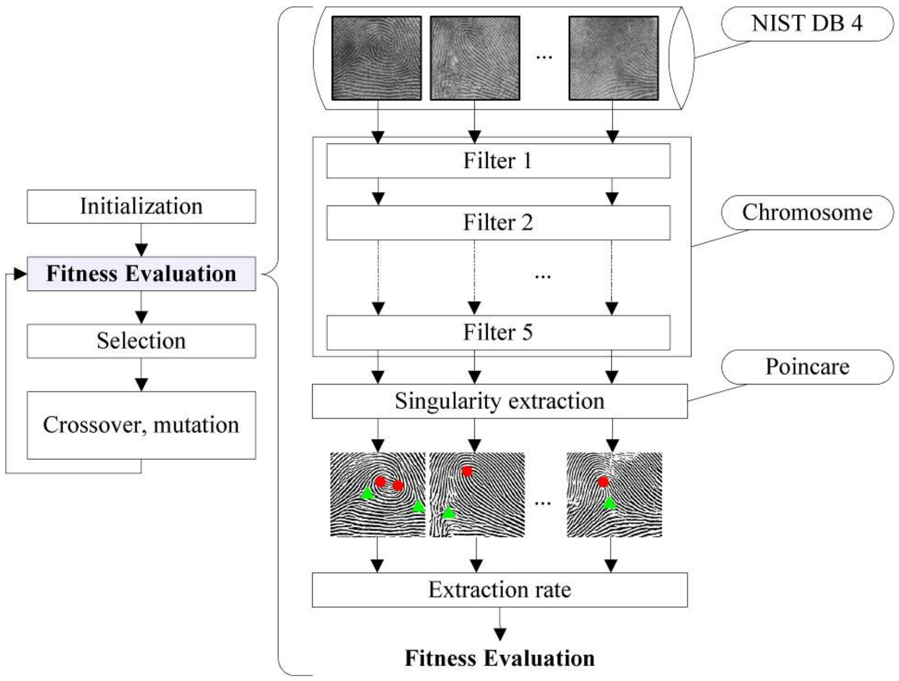


Fig. 3. The process of evaluating fitness (fitness function)

Table 1. The effect of various image filter

Group	Filter	Type	Effect	Number
Histogram	Brightness	3 values	Brightness value control	1~3
	Contrast	3 values	Contrast value control	4~6
	Stretch	-	Stretching histogram of images	7
	Equalize	-	Equalization histogram of images	8
	Logarithm	-	Logarithm histogram of images	9
Mask	Blur	6 masks	Smoothing images	10~15
	Sharper	4 masks	Sharpen images	16~19
	Median	10 masks	Noise elimination	20~29
Morphology	Erosion	10 masks	Elimination of single-pixel bright spots from images	30~39
	Dilation	10 masks	Elimination of single-pixel dark spots from images	40~49
	Opening	10 masks	Clean up images with noise	50~59
	Closing	10 masks	Clean up images with object holes	60~69
Fingerprint	Gabor	-	Ridge amplification with orientation field	70
None				0

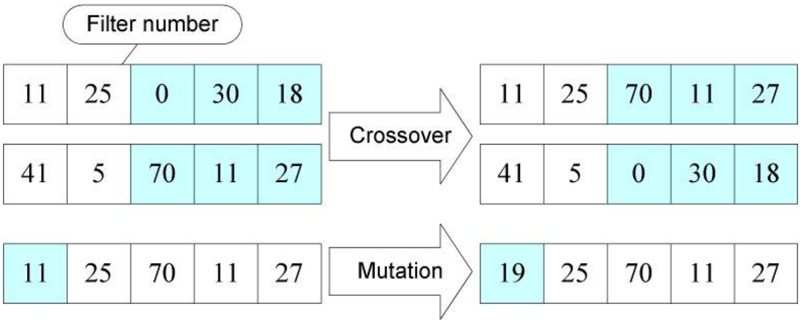


Fig. 4. The structure of chromosome and genetic operators

- Paired singularity (p): A set of the singularity points that s_d and s_e are paired if s_d is located within the tolerance (20 pixels) centered around s_e.
- Missing singularity (a): A set of the points that are located within the tolerance distance from singularity s_e but not singularity s_d, which means that the singularity extraction algorithm cannot detect the point.
- Spurious singularity (b): A set of the points that are located within the tolerance distance from singularity s_d but not singularity s_e, which is detected by the singularity extraction algorithm, but not real singularity.

The missing rate of singularity is estimated by the equation (1), the spurious rate is estimated by the equation (2), and the accuracy rate is estimated by the equation (3) with N samples.

$$P(a) = \frac{\sum_{i=1}^{N} n(a_i)}{\sum_{i=1}^{N} n(S_{ei})} \tag{1}$$

$$P(b) = \frac{\sum_{i=1}^{N} n(b_i)}{\sum_{i=1}^{N} n(S_{di})} \tag{2}$$

$$P(p) = 1 - \frac{\sum_{i=1}^{N} (n(a_i) + n(b_i))}{\sum_{i=1}^{N} (n(S_{di}) + n(S_{ei}))} \tag{3}$$

The accuracy of singularity is used as the fitness function of the genetic algorithm, where an individual that shows better enhancement performance obtains a higher score.

4 Experiments

The NIST Special Database 4 was used [21] to verify the proposed method. The NIST DB 4 consists of 4000 fingerprint images (2000 pairs). In the experiments, the training set was composed of the first 2000 images, f0001~f2000, and the test set consisted of the other 2000 images, s0001~s2000. In the database, 7308 singularities are manually marked by human experts, including 3665 on the training set and 3642 on the test set.

4.1 Analysis of the Process of Evolution

Table 2 shows the initial values of parameters in the experiment. The 40th generation results in a rise of 0.02 for the maximum fitness and a rise of 0.3 for the average fitness of the population which is 30 individuals. Fig. 5 shows the change of maximum and average fitnesses in each generation, where the maximum fitness increases steadily and the average fitness also shows a rise.

Table 2. The initial values of parameters

Parameter	Value
Generation	40
Population	30
Chromosome length	5
Selection rate	0.7
Crossover rate	0.7
Mutation rate	0.05
Elitist-strategy	Yes

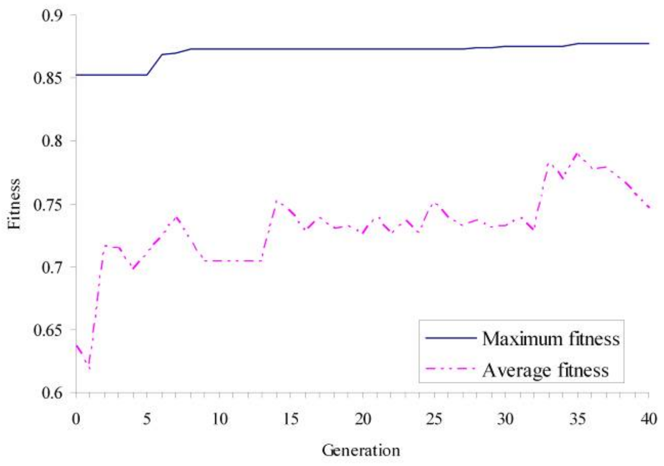


Fig. 5. The maximum and average fitnesses in each generation

Table 3. The number and type of image filters used in constructing a set of filters

generation	Filter #, type									
0	11	Lowpass 3×3 #2	25	Median 3×3 #3	0	NULL	30	Erosion 3×3 #1	18	Highpass 3×3 #3
6	11	Lowpass 3×3 #2	25	Median 3×3 #3	70	Gabor filter	11	Lowpass 3×3 #2	27	Median 1×3
7	19	Highpass 3×3 #4	25	Median 3×3 #3	70	Gabor filter	11	Lowpass 3×3 #2	27	Median 1×3
8	19	Highpass 3×3 #4	25	Median 3×3 #3	70	Gabor filter	11	Lowpass 3×3 #2	46	Dilation 5×5 #4
28	14	Gaussian 3×3	25	Median 3×3 #3	70	Gabor filter	70	Gabor filter	68	Closing 1×3
30	14	Gaussian 3×3	25	Median 3×3 #3	70	Gabor filter	70	Gabor filter	12	Lowpass 5×5
35	14	Gaussian 3×3	25	Median 3×3 #3	70	Gabor filter	70	Gabor filter	48	Dilation 3×3

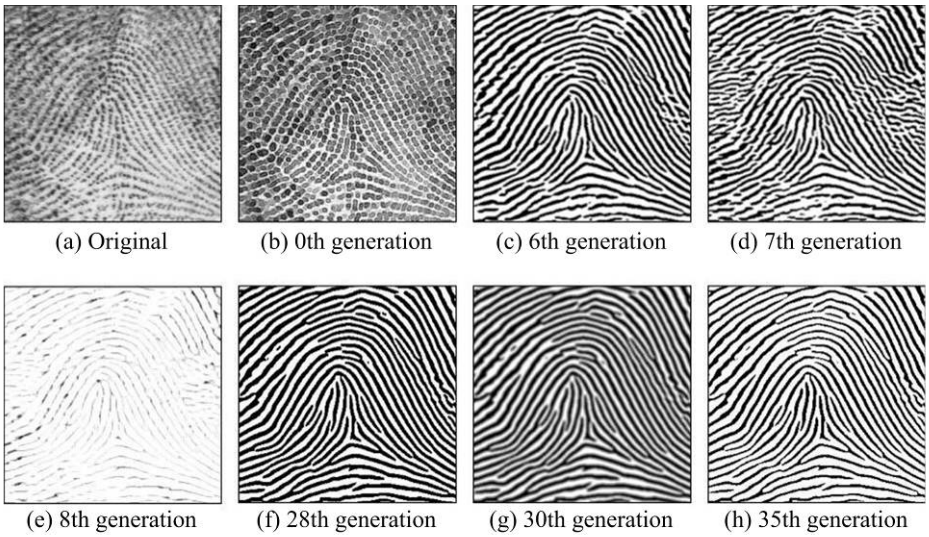


Fig. 6. Results of evolutionary filters

Table 3 shows filters obtained through the evolution. The Gabor filter is included in all the filters except for that of randomly initialized. These obtained filters are divided into the front part and rear part by the Gabor filter. The Gabor filter has the effect of ridge amplification with the orientation field, and orientation field correctly extracted can maximize the performance singularity extraction. Filters of the front part usually concern in the orientation field, and filters of the rear part effect an improvement for the result image of the Gabor filter. Finally, filter is obtained, which composes of Gaussian filter, median filter, two Gabor filters, and dilation. The input image is smoothed by the Gaussian filter, and impulse noise spikes of the image are removed by the median filter. The Gabor filter effects to correctly calculate the orientation field with ridge amplification. The dilation operation is used to remove small anomalies, such as single-pixel holes in objects and single-pixel-wide gaps of the image. It also has effect of thinning the ridges of images through the wider valleys of image. Fig. 6 shows example images obtained after applying filters.

4.2 Singularity Extraction

In the experiments, the error rate of singularity extraction over all individual filters was investigated, and we obtained 5 good filters that were 'median 3×3 rectangle mask filter,' 'closing 3×3 X mask operator,' 'Gaussian 3×3 mask filter,' 'Gabor filter,' and 'closing 3×3 diamond mask operator.' The heuristic filter is composed of these 5 filters, and the heuristic filter II has the reverse order of the previous one. Fig. 7 shows the comparison of various filters. The error rate of original was 18.2%, whereas individual

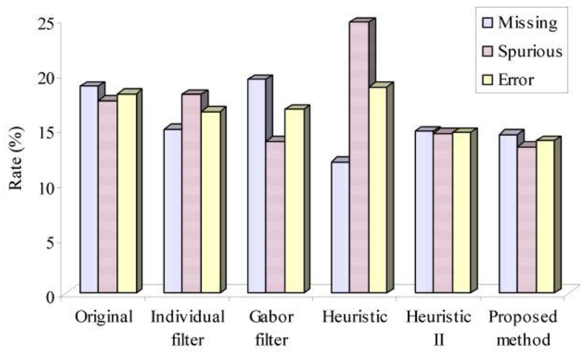


Fig. 7. The performance comparison with other filters

Table 4. Type and error rate of other filters

Filter	Filter type					Error rate
Original	NULL					18.2%
Individual	Closing Diamond 3×3					16.6%
Gabor	Gabor Filter					16.7%
Heuristic	Median Rectangle 3×3	Closing X 3×3	Gaussian 3×3	Gabor Filter	Closing Diamond 3×3	18.8%
Heuristic II	Closing Diamond 3×3	Gabor Filter	Gaussian 3×3	Closing X 3×3	Median Rectangle 3×3	14.7%
Proposed method	Gaussian 3×3	Median X 3×3	Gabor Filter	Gabor Filter	Dilation 1×3	13.9%

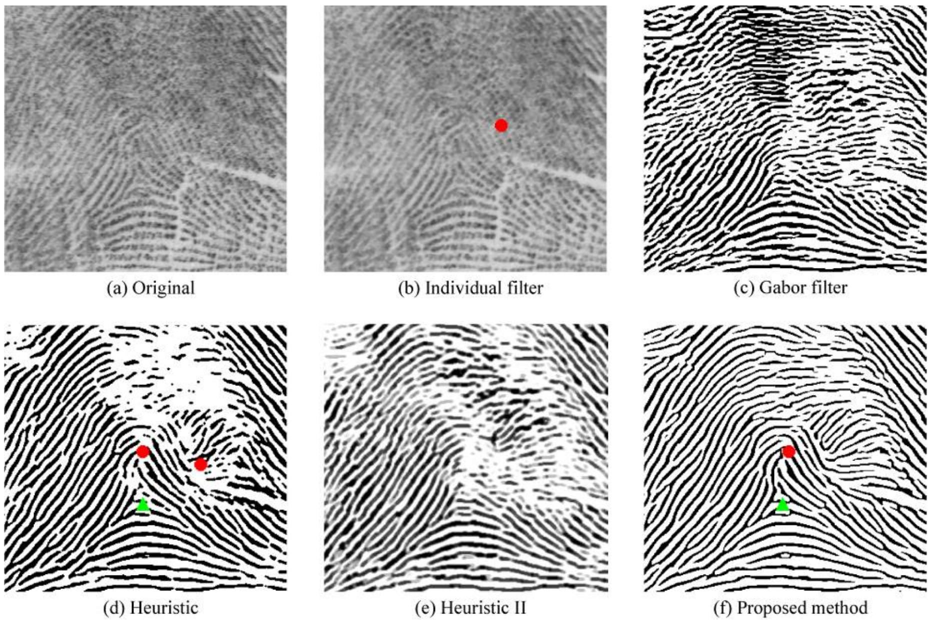


Fig. 8. Examples of various filters (● = core, ▲ = delta)

filters and the Gabor filter produced 16.6% and 16.7% error rates, respectively. The heuristic II filter yielded 14.7% error rate, while the proposed method obtained an error rate of 13.9%. The proposed method shows better performance in singularity extraction than the filters designed heuristically (Table 4).

The fingerprint image in Fig. 8 has one core and one delta. Fig. 8(a) shows the original image and singularity is not detected by the extraction algorithm. Individual filter that is a closing operation eliminates single-pixel dark spots from the image and smoothes it, but the extraction algorithm detects a spurious singularity because of unclear orientation on upper right-hand side field (Fig. 8(b)). The image is not enhanced well with the Gabor filter because the orientation field is not good (Fig. 8(c)). The extraction algorithm detects not all singularity but spurious singularity with the heuristic filter because of many blank spaces (Fig. 8(d)). The extraction algorithm with the proposed method detects no missing and spurious singularity (Fig. 8(f)).

5 Conclusions

It is important to detect singularities, which are special features of fingerprint, but limited in quality of fingerprint images. Image enhancement is required for correct extraction. In this paper, a genetic algorithm is used to obtain a combination of various individual filters for better performance in singularity extraction. In the experiments of NIST DB 4, the filter obtained shows better performance than the other filters.

In future work, we would like to apply the proposed method to the other fingerprint image databases. By changing the fitness function of the genetic algorithm, the method for better performance of identification and classification of fingerprints will be also investigated. Since the proposed method does not need any expert knowledge, we can also apply the method to various fields of image processing.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Biometrics Engineering Research Center (BERC) at Yonsei University.

References

[1] N. Ratha, and R. Bolle, *Automatic Fingerprint Recognition Systems*, Springer-Verlag, New York, 2004.

[2] A. K. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 302-314, 1997.

[3] A. M. Bazen, and S. H. Gerez, "Systematic methods for the computation of the directional fields and singular points of fingerprints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 905-919, 2002.

[4] A. M. Bazen, and S. H. Gerez, "Segmentation of fingerprint images," *In Proc. ProRISC2001 Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, pp. 276-280, 2001.

[5] L. Hong, Y. Wan, and A. Jain, "Fingerprint image enhancement: Algorithm and performance evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777-789, 1998.

[6] J. Yang, L. Liu, T. Jiang, and Y. Fan, "A modified Gabor filter design method for fingerprint image enhancement," *Pattern Recognition Letters*, vol. 24, pp. 1805-1817, 2003.

[7] S. Greenberg, M. Aldadjem, and D. Kogan, "Fingerprint image enhancement using filtering techniques," *Real-Time Imaging*, vol. 8, pp. 227-236, 2002.

[8] E. Zhu, J. Yin, and G. Zhang, "Fingerprint enhancement using circular Gabor filter," *International Conference on Image Analysis and Recognition*, pp. 750-758, 2004.

[9] D. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[10] L. M. Schmitt, "Theory of genetic algorithms," *Theoretical Computer Science*, vol. 259, pp. 1-61, 2001.

[11] S. B. Cho, "Emotional image and musical information retrieval with interactive genetic algorithm," *Proceedings of the IEEE*, vol. 92, no.4, pp. 702-711, 2004.

[12] N. R. Harvey, and S. Marshall, "The use of genetic algorithms in morphological filter design," *Signal Processing: Image Communication*, vol. 8, pp. 55-71, 1996.

[13] N. Yager, and A. Admin, "Fingerprint classification: A review," *Pattern Analysis and Application*, vol. 7, pp. 77-93, 2004.

[14] M. M. S. Chong, T. H. Ngee, L. Jun, and R. K. L. Gay, "Geometric framework for fingerprint image classification," *Pattern Recognition*, vol. 30, pp. 1475-1488, 1997.

[15] Q. Zhang, and H. Yan, "Fingerprint classification based on extraction and analysis of singularities and pseudo ridges," *Pattern Recognition*, vol. 37, pp. 2233-2243, 2004.

[16] A. K. Jain, S. Prabhakar, and L. Hong, “A multichannel approach to fingerprint classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 348-359, 1999.

[17] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni, “Fingerprint classification by directional image partitioning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 402-421, 1999.

[18] A. Ross, A. Jain, and J. Reisman, "A hybrid fingerprint matcher," *Pattern Recognition*, vol 36, pp.1661-1673, 2003.

[19] R. Gonzalez, and R. Woods, *Digital Image Processing*, Addison Wesley, Reading, MA. 1992.

[20] L. M. Schmitt, “Theory of genetic algorithms II: Models for genetic operators over the string-tensor representation to populations and convergence to global optima for arbitrary fitness function under scaling,” *Theoretical Computer Science*, vol. 310, pp. 181-231, 2004.

[21] C. Watson, and C. Wilson, *NIST Special Database 4. Fingerprint Database*. National Institute of Standard and Technology, 1992.

Evolutionary Generation of Prototypes for a Learning Vector Quantization Classifier

L.P. Cordella[1], C. De Stefano[2], F. Fontanella[1], and A. Marcelli[3]

[1] Dipartimento di Informatica e Sistemistica,
Università di Napoli Federico II,
Via Claudio, Napoli 21 80125 – Italy
{cordel, frfontan}@unina.it

[2] Dipartimento di Automazione, Elettromagnetismo,
Ingegneria dell'Informazione e Matematica Industriale,
Università di Cassino,
Via G. Di Biasio, Cassino (FR) 43 02043 – Italy
destefano@unicas.it

[3] Dipartimento di Ingegneria dell'Informazione e Ingegneria Elettrica,
Università di Salerno,
Fisciano (SA) 84084 – Italy
amarcelli@unisa.it

Abstract. An evolutionary computation based algorithm for data classification is presented. The proposed algorithm refers to the learning vector quantization paradigm and is able to evolve sets of points in the feature space in order to find the class prototypes. The more remarkable feature of the devised approach is its ability to discover the right number of prototypes needed to perform the classification task without requiring any a priori knowledge on the properties of the data analyzed. The effectiveness of the approach has been tested on satellite images and the obtained results have been compared with those obtained by using other classifiers.

1 Introduction

Classification problems are probably among the most studied ones in the field of computer science since its beginnings [1]. Classification is a process according to which an object is attributed to one of a finite set of classes or, in other words, it is recognized as belonging to a set of equal or similar entities, identified by a name (*label* in the classification field jargon). In the last decades, Evolutionary Algorithms (EAs) have demonstrated their ability to solve hard non linear problems characterized by very complex search spaces [2]; they have also been used to solve classification problems.

Genetic Algorithms (GAs), for example, have been widely applied for evolving sets of rules that predict the class of a given object. According the GAs-based approach, referred to as *learning classifier systems* (LCS) [3], the individuals in the population encode one or more prediction rules of the form IF-THEN, where the antecedent part of the rule contains a combination of conditions on some attributes of the patterns to be classified, while the consequent part rule expresses the class predicted by the rule. GAs for rule discovery can be divided into two main classes, depending on how rules are encoded by individuals. The classes are referred to as Michigan [4, 5] and Pittsburgh [6, 7]. The former approach employs a single individual for encoding one prediction rule, whereas in the latter approach each individual encodes a whole set of rules. Both these approaches were originally devised to solve single–class problems. Successively, also multi–class problems have been tackled. These kind of problems have been faced by introducing multiple populations, so that each population evolves the rule of a specific class. However, in this way it is hard to model the interaction between the rules representing different classes when new and unknown patterns have to be recognized. Moreover, another problem that affects the LCS algorithms is due to the difficulty of solving the conflict that comes up when single pattern is matched by several rules, each predicting a different class. As regards the GP–based methodologies, only recently they have been proposed to solve classification problems [8, 9, 10]. In [9], for example, a GP system has been devised to evolve equations involving simple arithmetic operators and feature variables, for hyper-spectral image classification. In [8], a GP approach has been employed for image classification problems, adding exponential functions, conditional functions and constants to the simple arithmetic operators. GP has also been used to evolve sets of fuzzy rules [11]. In [10], an interesting method, which considers a n-class problem as a set of two-class problems, has been introduced. When the expression for a particular class is searched, that class is considered as target, while the other ones are merged and treated as a single undesired class. In all approaches mentioned above, the number of prototypes to be used to solve the classification problem at hand is set exactly equal to the number n of classes to be dealt. Consequently, these approaches do not consider the existence of subclasses within one or more of the classes in the analyzed data set: in practice, each of these subclasses needs a specific prototype to be represented.

We propose a new classification method, based on the concept of Learning Vector Quantization (LVQ). Given a data set for which the patterns are feature vectors, this algorithm provides a set of *reference vectors*, to be used as prototypes for the classification task to be performed. The effect of any LVQ–based algorithm is a Voronoi tessellation of the feature space in which the data are represented. In practice, LVQ allows one to partition the feature space into a number of regions, each identified by one of the vectors provided after the training phase. Such regions are bordered by hyperplanes defined as the loci of points that are equidistant from the two nearest reference vectors. The reference vectors represent the prototypes of the points inside the regions and each region represents a cluster. The classification method proposed here uses a specifically devised evolutionary algorithm for evolving variable size sets of feature vectors. The remarkable feature of our approach is that it does not require any a priori knowledge about the actual number of prototypes, i.e. reference vectors, each class needs to be represented.

The proposed classification method has been tested on satellite images and the results obtained have been compared with those obtained by other well known classification techniques, included a standard LVQ. The data set at hand has been divided in a training and a test set. The former set was used to train both our system and the other classifiers taken into account for the comparison. As regards our system, once a set of prototypes has been obtained, the classification of an unknown patterns of the test set is performed by assigning to it the label of the nearest prototype in the feature space. The experiments performed by using the data taken into account have shown very interesting results and have confirmed the effectiveness of the proposed approach.

The remainder of the paper is organized as follows: in section 2 a formalization of data classification is described; section 3 illustrates the standard LVQ algorithm, while in section 4 the proposed approach is detailed. In section 5 the experimental results are presented, while section 6 is devoted to the conclusions.

2 Data Classification

In the data classification context a set of objects to be analyzed is called *data set*, and each object is called *pattern* and represented by $\mathbf{X} = (x_1, \ldots, x_\ell)$ with $\mathbf{X} \in \mathbf{S}$, where $\mathbf{S}$ is the universe of all possible elements characterized by ℓ features and x_i denotes the i–th feature of the pattern. A data set with cardinality N_D is denoted by $\mathcal{D} = \{\mathbf{X}_1, \ldots, \mathbf{X}_{N_D}\}$ with $\mathcal{D} \subseteq \mathbf{S}$. The set $\mathcal{D}$ is said *labeled* if it exists a set of integers:

$$\Lambda = \{\lambda_1, \ldots, \lambda_{N_D}\} : \lambda_i \in [1, c]$$

The i–th element λ_i of Λ is said the *label* of the i–th pattern $\mathbf{X}_i$ of $\mathcal{D}$. We will say that the patterns of $\mathcal{D}$ can be grouped into c different classes. Moreover, given the pattern $\mathbf{X}_i$ and the label $\lambda_i = j$, we will say that $\mathbf{X}_i$ belongs to the j–th class.

Given a data set $\mathcal{D} = \{\mathbf{X}_1, \ldots, \mathbf{X}_{N_D}\}$ containing c classes, a classifier Γ is defined as a function

$$\Gamma : \mathcal{D} \longrightarrow [0, c]$$

In other words, a classifier assigns a label $\gamma_i \in [0, c]$ to each input pattern $\mathbf{X}_i$. If $\gamma_i = 0$, the corresponding pattern $\mathbf{X}_i$ is said *rejected.* This fact means that the classifier is unable to trace the pattern back to any class.

The pattern $\mathbf{X}_i$ is *recognized* by Γ if and only if:

$$\gamma_i = \lambda_i$$

otherwise the pattern is said *misclassified.* If N_{corr} is the number of patterns of $\mathcal{D}$ recognized by Γ the ratio N_{corr}/N_D is defined as the *recognition rate* of the classifier Γ obtained on the data set $\mathcal{D}$.

```
begin
  initialize the reference vectors $\omega_1, \ldots, \omega_k$;
  initialize the learning rate $a(0)$;
  while stop_condition is false do
    for $i = 0$ to $N_{\rm tr}$ do
      find $j$ so that $\|\mathbf{x}_i - \omega_j\| = \min$;
    end for
    update the vector $\omega_j$ as follows:
    if $\lambda_i = C_j$ then
      $\omega_j(\text{new}) = \omega_j(\text{old}) + a(\mathbf{x}_i - \omega_j(old))$
    end if
    if $\lambda_i \neq C_j$ then
      $\omega_j(\text{new}) = \omega_j(\text{old}) - a(\mathbf{x}_i - \omega_j(old))$
    end if
  end while
end
```

Fig. 1. The algorithm used in order to determine the position of the reference vectors in the feature space. $N_{\rm tr}$ is the number of patterns in the training set, while λ_i and C_j respectively represent the labels of the pattern $\mathbf{x}_i$ and of the "winner" vector ω_j. This algorithm is often referred in the literature to as "the winner take all" (WTA). Note that usually the stop_condition specifies a certain number of iterations (*epochs* in the LVQ jargon).

3 Learning Vector Quantization

The Learning Vector Quantization (LVQ) methodology was introduced by Kohonen [12] and in the last years has widely been used to perform classification tasks. LVQ classifiers have been applied in a variety of practical problems, such as medical image analysis, classification of satellite spectral data, fault detection in technical processes, and language recognition [13]. The LVQ approach offers a method to form a quantized approximation of an input data set $\mathcal{D} \subset R^p$ using a finite number k of *reference vectors* $\omega_i \in R, i = 1, \ldots, k$. After that an LVQ has been trained, the feature space in which the data are represented, is partitioned with a Voronoi tessellation. This tessellation partitions the feature space into a certain number of disjoint regions, each identified by a different reference vector. These regions are bordered by hyperplanes defined as the loci of points that are equidistant from the two nearest reference vectors. Hence, a reference vector represents the prototype of the points inside the corresponding regions.

Given an incoming pattern $\mathbf{x}$ to be recognized and a set of reference vectors $\{\omega_1, \ldots, \omega_k\}$, the classification is performed by computing the Euclidean distance between $\mathbf{x}$ and each of the reference vector ω_i. The pattern is then assigned to the closest reference vector and is recognized as belonging to the same class of the reference vector to which it has been assigned. In the LVQ method the reference vectors are a priori labeled and algorithmically determined by using a

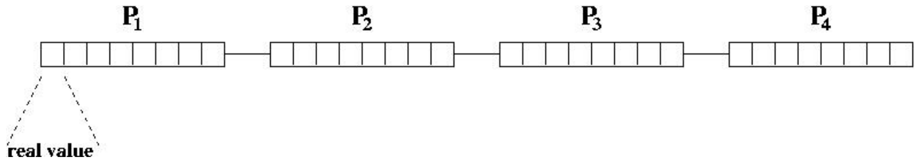


Fig. 2. An example of individual containing 4 prototypes. In this example, the prototypes are vectors in a 8-dimensional space.

training set of labeled patterns (see figure 1). Note that, usually, the number of reference vectors for each of the classes have to be provided by the user.

4 Prototype Generation

As mentioned above, the prototypes to be used for classification are represented by points in the feature space. The method proposed in this paper for finding a good set of prototypes is based on a particular class of genetic algorithms [2], namely the Breeder Genetic Algorithms (BGA) [14], in which the individuals are encoded as real valued vectors. In our case, an individual consists of a variable length list of feature vectors, each one representing a prototype (see Figure 2).

The system, accordingly to the EA paradigm, starts by generating a population of $\mathcal{P}$ individuals. The number of prototypes (in the following referred as *length*) of these initial individuals is randomly assigned in the range $[N_{\min}, N_{\max}]$. Each prototype is initialized by randomly choosing a pattern in the training set. Afterwards, the fitness of the individuals generated is evaluated. A new population is generated in two ways: on one side, according to an elitist strategy, the best $\mathcal{E}$ individuals are selected and just copied. On the other side, $(\mathcal{P}-\mathcal{E})/2$ couples of individuals are selected by using a selection mechanism. The crossover operator is then applied to each of the selected couples, according to a chosen probability factor p_c. The mutation operator is then applied to the individuals according to a probability factor p_m. Finally the obtained individuals are added to the new population. The process just described is repeated for N_g generations.

The fitness function, the selection mechanism and the operators employed are described in the following.

4.1 Fitness Function

Each individual is evaluated using a training set $\mathcal{D}_{tr}$ containing N_{tr} patterns. This evaluation implies the following steps:

1. Every pattern in the training set is assigned to the nearest prototype (i.e. reference vector) in the individual to be evaluated. Euclidean distance is used in the feature space. After this step, n_i ($n_i \geq 0$) patterns will have been assigned to the i-th prototype. The prototypes for which $n_i > 0$ will be referred in the following as *valid*. The prototypes for which $n_i = 0$ will be ignored in the following steps.

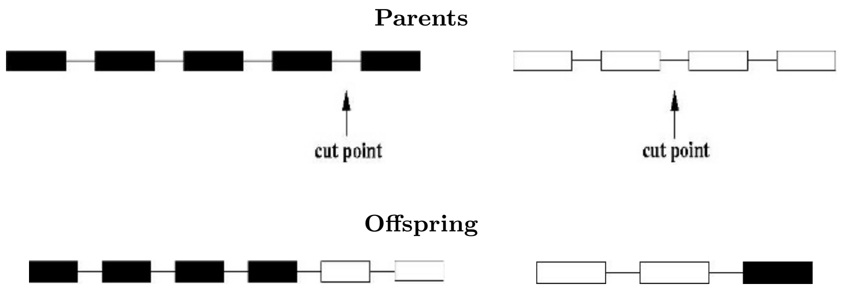


Fig. 3. An example of application of the crossover operator

2. Each valid prototype of an individual is labeled with the label most widely represented in the corresponding cluster.
3. The recognition rate obtained on $\mathcal{D}_{tr}$ is computed and assigned as fitness value to the individual.

In order to favor the individuals able to obtain good performances with a lesser number of prototypes, the fitness of each individual is increased by $0.1/N_p$, where N_p is the number of prototypes in an individual. Moreover, the individuals having a number of prototypes out of the interval $[N_{\min}, N_{\max}]$ are *killed*, i.e. marked in such a way that they are not chosen by the selection mechanism.

4.2 Selection Mechanism

The tournament method has been chosen as selection mechanism. In the tournament selection, a number $\mathcal{T}$ of individuals is chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated for as many individuals have to be chosen. Such a mechanism ensures to control the loss of diversity and the selection intensity [15].

4.3 Genetic Operators

In the approach presented here two genetic operators have been devised: *crossover* and *mutation*. The crossover operator belongs to the wider class of recombination operators: it accepts in input two individuals and it yields as output two new individuals. This operator acts at list level and gives our system the important feature of automatically discovering the number of prototypes actually needed to represent the classes defined in the problem at hand. The mutation operator, instead, manipulates a single individual. Its effect is that of "moving" the prototypes (i.e. vectors) of the input individual in the feature space. These operators are detailed in the following.

Crossover. The crossover operator is applied to two individuals I_1 and I_2 and yields two new individuals by swapping parts of the lists of the initial individuals, without breaking any single prototype. Assuming that the length of I_1 and I_2

```
begin
  for $j = 0$ to $N_F$ do
    range = rndreal(0.1 * $\delta_j$)
    if flip($p_m$) then
      $\mathbf{x}[j] = \mathbf{x}[j] \pm$ range (+ or - with equal probability);
    end if
  end for
end
```

Fig. 4. The mutation operator applied to each of the prototypes, i.e. reference vectors, in an individual. N_F is the number of features of the patterns in the data analyzed. δ_j is the range of the j-th feature computed on the training set, while p_m represents the probability of mutation of each single feature value in the prototype.

are respectively l_1 and l_2, the crossover is applied in the following way: the first individual is split in two parts by randomly choosing an integer t_1 in the interval $[1, l_1]$. The obtained lists of vectors $I_1^{'}$ and $I_1^{''}$ will have length t_1 and $l_1 - t_1$ respectively. Analogously, by randomly choosing an integer t_2 in the interval $[1, l_2]$, two lists of prototypes $I_2^{'}$ and $I_2^{''}$, respectively of length t_2 and $l_2 - t_2$, are obtained from I_2. At this stage, in order to obtain a new individual, the lists $I_1^{'}$ and $I_2^{''}$ are merged. This operation yields a new individual of length $t_1 + l_2 - t_2$. The same operation is applied to the remaining lists $I_2^{'}$ and $I_1^{''}$ and a new individual of length $t_2 + l_1 - t_1$ is obtained. The number of the swapped prototypes depends on the integers t_1 and t_2. An example of application of this operator is given in figure 3. As mentioned above the implemented crossover operator allows one to obtain offspring individuals whose length may be quite different from that of the parents. As a consequence, during the evolution process, individuals made of a variable number of prototypes can be evolved.

Mutation. Given an individual I, the mutation operator is independently applied to each prototype of I. In figure 4 the algorithm used to modify each of the prototypes in an individual is shown.

5 Experimental Results

In order to ascertain its effectiveness, the proposed approach has been tested on data extracted from two landsat 6 band multispectral images. The first one[1] is 2030x1167 pixels large and has been taken in order to distinguish between forest–non forest areas, while the second one[2] is a 1000x1000 pixels large, related to the land cover mapping for desertification studies. To this images a segmentation method has been applied in order to obtain regions formed by the same type of pixel [16]. For each of the region provided by the segmentation, a set of features have been extracted, related to its geometrical characteristics and to its spectral

[1] The image has been provided by courtesy from JRC.

[2] This image has been provided by courtesy from ACS spa as part of the Desert Watch project.

data. For each region the extracted features have been used to build up a data record.

From the first image a data set made up of 2500 items, i.e. regions, have been derived; each item may belong to one of two classes, forest or non–forest. From the second image 7600 items have been extracted and the items may belong to 7 classes, representing various land cover types: various vegetation or water. Although the segmentation process used extracts more than 10 features for each of the regions identified, for both the images analyzed only six features have been considered.

Preliminary trials have been performed to set the basic evolutionary parameters reported in Table 1. This set of parameters has been used for all the experiments reported in the following. Since our approach is stochastic, as well as all the EC–based algorithms, 20 runs have been performed for each data set taken into account. The reported results are those obtained using the individual having the highest fitness among those obtained during the 20 performed runs.

The results obtained by our method on the data described above have been compared with those obtained by other three classification algorithms: nearest–neighbor (NN), k–NN and a standard LVQ. These algorithms are detailed in the following:

LVQ. The LVQ used for the comparison of our results is an improved version of that described in section 3, it is called *Frequency Sensitive Competitive Learning* (FSCL) [17] and is often used to compare the performances of other algorithms.

Nearest Neighbor. Let $\mathcal{D}_{\text{tr}}$ be a training set of N_{tr} labeled patterns represented by feature vectors. A nearest neighbor (NN) classifier recognizes an unknown pattern $\mathbf{x}$ by computing the Euclidean distance between $\mathbf{x}$ and each of the patterns in $\mathcal{D}_{\text{tr}}$. Then $\mathbf{x}$ is recognized as belonging to the same class of the nearest pattern in $\mathcal{D}_{\text{tr}}$. It has been shown that a NN classifier does not guarantee the minimum possible error, i.e. the Bayes rate [1].

***k*-Nearest Neighbor.** A k-Nearest Neighbor (k-NN) classifier is an extension of the NN classifier described above. In fact, given a training set $\mathcal{D}_{\text{tr}}$, a pattern $\mathbf{x}$ to be recognized is attributed to the most widely represented class among the k nearest patterns of $\mathcal{D}_{\text{tr}}$. In the experiments reported in the following the number of neighbors has been varied between 1 and 15.

Table 1. Values of the basic evolutionary parameters used in the experiments

Parameter	symbol	value
Population size	$\mathcal{P}$	300
Tournament size	$\mathcal{T}$	6
elithism size	$\mathcal{E}$	5
Crossover probability	p_c	0.4
Mutation probability	p_m	0.05
Number of Generations	N_g	500

Table 2. Means and standard deviations of recognition rate on the test set for the forest cover dataset

	NN	**9-NN**	**LVQ**	**EC-LVQ**
Mean	72.69	83.05	72.05	82.50
Std	4.34	0.25	3.36	1.7
$\mathbf{N_P}$	2250	2250	80	10.6 (1.7)

Table 3. The recognition rates on the test set obtained for the land cover dataset

	NN	**6-NN**	**LVQ** (700)	**EC-LVQ**
Best	69.2	74.08	76.47	75.6
$\mathbf{N_P}$	3800	3800	700	136

5.1 Comparison Findings

In Table 2 the results obtained on the forest cover data set are shown. In all the runs performed on this data set the minimum length N_{min} allowed for an individual has been set to 2, while the maximum one has been set to 20. In order to avoid any bias in the comparison, due to the low number of patterns in the data set, the 10 fold cross validation procedure has been adopted. In this procedure, the performances of a classifier on a data set $\mathcal{D}$ are evaluated by randomly dividing $\mathcal{D}$ into 10 disjoint sets of equal size $N/10$, where N is the total number of patterns in $\mathcal{D}$. Then the classifier is trained 10 times, each time with a different set held out as a test set. In order to evaluate the performance of the classifier on unknown data, the performance is computed as the mean of the results obtained on the ten different test sets [1] As a consequence of the choice of this procedure, 200 runs have been performed.

In Table 2, the mean and the standard deviation obtained on the 10 test sets are reported together with the number of prototypes employed are reported. As regards the NN and the k–NN classifiers, the number of prototypes equals the number of patterns in the training set, while for the LVQ this number has been set to 80. For our EC–based LVQ method, this number is not provided by the user, but it has been automatically found by the implemented system. Particularly, the average number of prototypes, found for the ten considered test set, equals to 10.6, while the standard deviation is 1.7. Thus the automatism devised allow the system to strongly reduce the number of prototypes needed to perform the classification task demanded. The outcome of this reduction is a strong enhancement of the classifier efficiency. As regards the recognition rate obtained on test set by the different algorithms, the performance of our system is significantly better than that of the NN and LVQ classifiers, although that of the k-NN (obtained setting k equal to 9) are slightly better than that of our

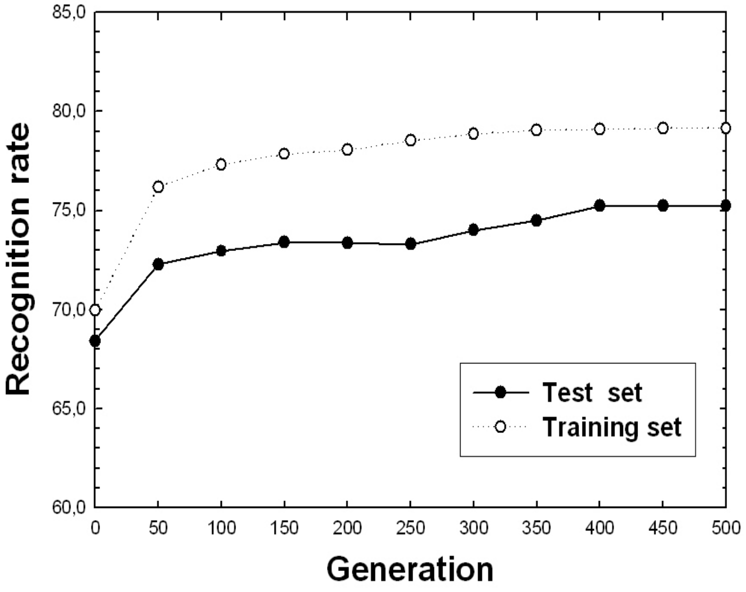


Fig. 5. Recognition rate on training and test sets for the land cover data set during the best run

system. However, in our opinion, the huge difference in the number of necessary prototypes used compensates this little difference in the recognition rate.

In Table 3 the results obtained on the land cover data set are shown. In this case the original data set has been randomly split in two sets, respectively as training set and test set. For this data set the total number of executed runs has been 20. In Table 3 the best results obtained and the number of prototypes employed are reported. Our results are significantly better than those obtained form the NN classifier and slightly better than that obtained from the k–NN (in this case k has been set equal to 6). Only the LVQ classifier has obtained a performance slightly better than ours, but such performance has been obtained with a total number of 700 prototypes, while our performance has been obtained using only 136 prototypes. Also in this case, the difference in the number of prototypes compensates the little difference in the performance achieved.

In a learning process, in most cases, when the maximum performance is achieved on training set, the generalization power, i.e. the ability of obtaining similar performance on unknown data (the test set), may significantly decrease. In order to investigate such aspect for our system, the recognition rates on training and test set have been taken into account for the different considered data sets. In Figure 5 such recognition rates, evaluated every 50 generations, in the best run for the desert data set, are displayed. It can be observed from the figure that, in the experiments carried out, the recognition rate increases with the number of generations both for the training set and for the test set. The best recognition rates occur in both cases nearby generation 400. Moreover, the fact that the difference between the two recognition rates does not tend to increase when that on the training set reaches its maximum, demonstrates the good generalization power of our system.

6 Conclusions and Feature Work

A new method that uses the evolutionary computation paradigm has been devised for generating prototypes for a LVQ–based classifier. The patterns belonging to the different classes, represented as vectors in a feature space, are represented by prototypes obtained by evolving an initial population of randomly selected feature vectors. The devised approach does not require any a priori knowledge about the actual number of prototypes needed to represent the classes defined in the problem at hand. The method has been tested on satellite images and the results have been compared with those obtained by other classification methods. The obtained results and the comparisons performed have confirmed the effectiveness of the approach and outlined the good generalization power of the proposed method.

The results could be easily improved by applying the mutation operator in a way that takes into account the performances obtained by the single prototypes. In practice, the probability of application of the mutation operator to a single prototype should be computed as a function of its performance. Specifically, the lower is the recognition rate obtained by the prototype, the lower should be the probability of applying of the mutation to it. In this way, the research of prototypes becomes more effective, since the probability of modifying "good" prototypes is much lower than that of modifying "bad" prototypes, i.e. those performing worse in recognizing patterns belonging to the same class.

Acknowledgments

The authors gratefully acknowledge the support of the Joint Research Centre (JRC) that supplied the forest cover data and the ACS spa which supplied the land mapping data as part of the Desert Watch project.

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & sons, Inc. (2001)
2. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley (1989)
3. Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Learning Classifier Systems: From Foundations to Applications. Volume 1813 of Lecture Notes on Artificial Intelligence. Springer-Verlag, Berlin, Germany (2000)
4. Giordana, A., Neri, F.: Search-intensive concept induction. Evolutionary Computation **3** (1995) 375–416

5. Greene, D.P., Smith, S.F.: Competition-based induction of decision models from examples. Machine Learning (1993) 229–257
6. Janikow, C.Z.: A knowledge-intensive genetic algorithm for supervised learning. Machine Learning (1993) 189–228
7. De Jong, K.A., Spears, W.M., Gordon, D.F.C., Janikow, Z.: Using genetic algorithms for concept learning. Machine Learning (1993) 161–188
8. Agnelli, D., Bollini, A., Lombardi, L.: Image classification: an evolutionary approach. Pattern Recognition Letters **23** (2002) 303–309
9. Rauss, P.J., Daida, J.M., Chaudhary, S.A.: Classification of spectral image using genetic programming. In: Genetic and Evolutionary Computation Conference (GECCO). (2000) 726–733
10. Kishore, J.K., Patnaik, L.M., Mani, V., Agrawal, V.K.: Application of genetic programming for multicategory pattern classification. IEEE Transactions on Evolutionary Computation **4** (2000) 242–258
11. Mendes, R., Voznika, F., Freitas, A., Nievola, J.: Discovering fuzzy classification rules with genetic programming and co-evolution. In: Principles of Data Mining and Knowledge Discovery (Proc. 5th European Conference PKDD 2001) - Lecture Notes in Artificial Intelligence. 2168, Berlin, Springer-Verlag (2001) 314–325
12. Kohonen, T.: Self-Organizing Maps. 3-rd edn. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2001)
13. Karayiannis, N.B.: Learning vector quantization: A review. International Journal of Smart Engineering System Design **1** (1997) 33–58
14. Muhlenbein, H., Schlierkamp-Voosen, D.: The science of breeding and its application to the breeder genetic algorithm (bga). Evolutionary Computation **1** (1993) 335–360
15. Blickle, T., Thiele, L.: A comparison of selection schemes used in genetic algorithms. Technical Report 11, Swiss Federal Institute of Technology (ETH), Gloriastrasse 35, 8092 Zurich, Switzerland (1995)
16. D'Elia, C., Poggi, G., Scarpa, G.: A tree-structured random markov field model for bayesian image segmentation. IEEE Transactions on Image Processing **12** (2003) 1259–1273
17. Ahalt, S., Krishnamurthy, A., Chen, P., Melton, D.: Competitive learning algorithms for vector quantizationn. Neural Networks **3** (1990) 277–290

Automatic Classification of Handsegmented Image Parts with Differential Evolution

I. De Falco[1], A. Della Cioppa[2], and E. Tarantino[1]

[1] Institute of High Performance Computing and Networking, National Research Council of Italy (ICAR–CNR), Via P. Castellino 111, Naples 80131, Italy
{ivanoe.defalco, ernesto.tarantino}@na.icar.cnr.it

[2] Department of Computer Science and Electrical Engineering, University of Salerno, Via Ponte don Melillo 1, Fisciano (SA) 84084, Italy
adellacioppa@unisa.it

Abstract. Differential Evolution, a version of an Evolutionary Algorithm, is used to perform automatic classification of handsegmented image parts collected in a seven–class database. Our idea is to exploit it to find the positions of the class centroids in the search space such that for any class the average distance of instances belonging to that class from the relative class centroid is minimized. The performance of the resulting best individual is computed in terms of error rate on the testing set. Then, such a performance is compared against those of other ten classification techniques well known in literature. Results show the effectiveness of the approach in solving the classification task.

Keywords: Differential evolution, classification, centroids, multi–class database, recognition of image parts.

1 Introduction

Differential Evolution [1] [2] (DE) is a version of an Evolutionary Algorithm [3] [4] which uses vector differences for perturbing the population. In this paper we wish to examine its ability to effectively and efficiently perform automatic classification in a multi–class database.

A classification tool [5] [6] [7] is usually a part of a more general automatic pattern recognition system and aims at assigning class labels to the observations previously gathered by some sensor. To fulfil its task, a classifier relies on some features extracted in numeric or symbolic form by a feature extraction mechanism.

The classification scheme is usually based on the availability of a set of patterns that have already been classified or described (training set). In such a situation, termed *supervised*, starting from these presented data, the system has to guess a relation between the input patterns and the class labels and to generalize it to unseen patterns (testing set). Learning can also be *unsupervised*, in the sense that the system is not given an a *priori* labelling of patterns, instead it establishes the classes itself based on the statistical regularities of the patterns.

As far as we know, there exist no papers in literature in which DE is directly employed as a tool for supervised classification of multi–class database instances. Just one paper [8] analyzes the adaptation of DE and of other techniques in the fuzzy modelling context for the classification problem, while in [9] DE is used for proper weighting of similarity measures in the classification of high dimensional and large scale data. Moreover, a few papers deal with DE applied to unsupervised classification, for example on hard clustering problems [10] and on images [11]. Therefore, in this paper we aim to evaluate DE efficiency in performing a centroid–based supervised classification by taking into account a database composed by handsegmented parts of outdoor images, collected in a seven–class database. In the following, the term centroid means simply "class representative" and not necessarily "average point" of a cluster in the multidimensional space defined by the database dimensions. Our idea is to use DE to find the positions of the class centroids in the search space such that for any class the average distance of instances belonging to that class from the relative class centroid is minimized. Error percentage for classification on testing set is computed on the resulting best individual. Moreover, the results are compared against those achieved by ten well–known classification techniques.

Paper structure is as follows: Section 2 describes DE basic scheme, while Section 3 illustrates the application of our system based on DE and centroids to the classification problem. Section 4 reports on the database faced, the results achieved by our tool and the comparison against ten typical classification techniques. Finally Section 5 contains our conclusions and future works.

2 Differential Evolution

Differential Evolution (DE) is a stochastic, population-based optimization algorithm [1] [2]. It was firstly developed to optimize real parameters of a real–valued function and uses vectors of real numbers as representations of solutions.

The seminal idea of DE is that of using vector differences for perturbing the genotype of the individuals in the population. Basically, DE generates new individuals by adding the weighted difference vector between two population members to a third member. This can be seen as a non–uniform crossover that can take child vector parameters from one parent more often than it does from others. If the resulting trial vector yields a better objective function value than a predetermined population member, the newly generated vector replaces the vector with which it was compared. By using components of existing population members to construct trial vectors, recombination efficiently shuffles information about successful combinations, enabling the search for an optimum to focus on the most promising area of solution space.

In more detail, given a minimization problem with m real parameters, DE faces it starting with a randomly initialized population $P(t = 0)$ consisting of n individuals each made up by m real values. Then, the population is updated from a generation to the next one by means of some transformation. Many different transformation schemes have been defined. The authors [1] [2] tried to come up with a sensible naming–convention, so they decided to name any DE strategy with a string like *DE/x/y/z.* In it DE stands for Differential Evolution, x is a string which denotes the vector to be perturbed (best = the best individual in current population, rand = a randomly chosen one, rand–to–best = a random one, but the current best participates in the perturbation too), y is the number of difference vectors taken for perturbation of x (either 1 or 2), while z is the crossover method (exp = exponential, bin = binomial). We have decided to perturb a random individual by using one difference vector and by applying binomial crossover, so our strategy can be referenced as *DE/rand/1/bin.* In it for the generic i–th individual in the current population three integer numbers r_1, r_2 and r_3 in $[1, n]$ differing one another and different from i are randomly generated. Furthermore, another integer number k in the range $[1, m]$ is randomly chosen. Then, starting from the i–th individual a new trial one i' is generated whose generic j–th component is given by:

$$x_{i',j} = x_{r_3,j} + F \cdot (x_{r_1,j} - x_{r_2,j}) \tag{1}$$

provided that either a randomly generated real number ρ in $[0.0, 1.0]$ is lower than a value CR (parameter of the algorithm, in the same range as ρ) or the position j under account is exactly k. If neither is verified then a simple copy takes place: $x_{i',j} = x_{i,j}$. F is a real and constant factor in $[0.0, 1.0]$ which controls the magnitude of the differential variation $(x_{r_1,j} - x_{r_2,j})$, and is a parameter of the algorithm.

This new trial individual i' is compared against the i–th individual in current population and, if fitter, replaces it in the next population, otherwise the old one survives and is copied into the new population. This basic scheme is repeated for a maximum number of generations g.

DE pseudocode is shown in the following in **Algorithm 1**.

3 DE Applied to Classification

3.1 Encoding

We have chosen to face the classification task by using a tool in which DE is coupled with centroids mechanism (we shall hereinafter refer to it as DE–C system). Specifically, given a database with C classes and N attributes, DE–C should find the optimal positions of the C centroids in the N-dimensional space, i.e. it should determine for any centroid its N coordinates, each of which can take on, in general, real values. With these premises, the i-th individual of the population is encoded as it follows:

$$(\boldsymbol{p}_i^1, \ldots, \boldsymbol{p}_i^C) \tag{2}$$

where the position of the j–th centroid is constituted by N real numbers representing its N coordinates in the problem space:

$$\boldsymbol{p}_i^j = \{p_{1,i}^j, \ldots, p_{N,i}^j\} \tag{3}$$

Algorithm 1. DE Algorithm

```
begin
  randomly initialize population
  evaluate fitness $\psi$ of all individuals
  while (maximal number of generations $g$ is not reached) do
    begin
      for $i = 1$ to $n$ do
        begin
          choose three integer numbers $r_1$, $r_2$ and $r_3$ in $[1, n]$
                  differing each other and different from $i$
          choose an integer number $k$ in $[1, m]$
          for $j = 1$ to $m$ do
            begin
              choose a random real number $\rho$ in $[0.0, 1.0]$
              if
                $((\rho < CR)$ OR $(j = k))$
                  $x_{i',j} = x_{r_3,j} + F \cdot (x_{r_1,j} - x_{r_2,j})$
                else
                  $x_{i',j} = x_{i,j}$
            end
          if
            $\psi(x_{i',j}) < \psi(x_{i,j})$
              insert $x_{i',j}$ in the new population
            else
              insert $x_{i,j}$ in the new population
        end
    end
end
```

Then, any individual in the population consists of $C \cdot N$ components, each of which is represented by a real value.

3.2 Fitness

Following the classical approach to supervised classification, also in our case a database is divided into two sets, a training one and a testing one. The automatic tool learns on the former, and its performance is evaluated on the latter.

Our fitness function ψ is computed as the sum on all the training set instances of the euclidean distance in the N-dimensional space between the generic instance $\boldsymbol{x}_j$ and the centroid of the class CL it belongs to according to database $(\boldsymbol{p}_i^{CL_{\text{known}}(\boldsymbol{x}_j)})$. This sum is divided by D_{Train}, which is the number of instances composing the training set. In symbols, the fitness of the i–th individual is given by:

$$\psi(i) = \frac{1}{D_{\text{Train}}} \cdot \sum_{j=1}^{D_{\text{Train}}} d\big(\boldsymbol{x}_j, \boldsymbol{p}_i^{CL_{\text{known}}(\boldsymbol{x}_j)}\big) \tag{4}$$

When computing distance, any of its components in the N–dimensional space is normalized with respect to the maximal range in the dimension, and the sum of distance components is divided by N. With this choice, any distance ranges within $[0.0, 1.0]$, and so does ψ. Given the chosen fitness function, the problem becomes a typical minimization problem.

This measure has been preferred to the classical ones such as the percentage of incorrectly assigned instances in training set because this latter can vary with steps equal to D_{Train} only, whereas ours can do it with greater continuity. Moreover, our fitness varies for small variations in centroid positions too, which might be ininfluent in the other case, where small changes in centroid positions might not cause the transition of any element from a class to another, thus no variation in incorrectly classified instances percentage would be obtained. Of course, the underlying hypothesis, i.e. that reducing these fitness values reduces the number of incorrectly classified instances in the testing set, should be supported by experimental evidence.

Performance of a run, instead, is computed as the percentage $\%err$ of instances of testing set which are incorrectly classified by the best individual (in terms of the above fitness) achieved in the run. With this choice DE–C results can be directly compared to those provided by other classification techniques, which yield as output classification error percentage.

4 Experiments and Results

4.1 The Image Database

We have used the Image database which we downloaded from the UCI database repository [12], though actually it was created by the Vision Group at the University of Massachusetts and was donated to UCI in 1990. It was obtained by its authors starting from a set of seven RGB outdoor images representing exteriors of houses. One of those images is shown in Fig.1 (taken from [13]). Preliminarily each image was assigned area segments by hand to give an implicit label to each pixel. Then each square region composed by $3 \cdot 3$ neighboring pixels was taken into account: if and only if it was homogeneous, i.e. if the nine composing pixels had the same label, then this region was added as a new instance to an intermediate database, otherwise it was discarded. Finally, some of the items in the intermediate database were drawn randomly to create the final Image database.

The downloaded database is composed by a training set and a testing one. The former consists of 210 instances, divided into seven classes, one for each of seven image components: brickface, sky, foliage, cement, window, path and grass. Any class is represented in this training set by 30 instances. The testing set provided, instead, is very large and consists of 2,100 elements. From them we have chosen for each class its first 20 instances in their order of appearance in the testing set. Thus, our testing set is composed by $7 \cdot 20 = 140$ items and our whole database is made up by $210+140 = 350$ instances. As a result, the training set is assigned 60% of the database instances, and the testing set the remaining 40%, which are quite usual percentages in classification. The database contains

Fig. 1. One of the images used to create the database (taken from [13])

19 attributes some represented by integer values and some by real ones. All the attributes are listed in Table 1 together with their type and their minimal and maximal values. There are no missing values.

Since DE–C can handle real values while some attributes are of integer type, care has been posed in dealing with this issue. Namely, attributes with integer values have been converted to real values, and the reverse conversion with suitable rounding is carried out in the obtained solutions.

Some database features are displayed in Fig. 2. Its upper part shows the relationship between attributes 2 and 11, and reveals that in this case the instances are well grouped according to their class, so we might hypothesize that the database might be easily divided into classes without making many errors. Unfortunately this conclusion is not true, and a careful analysis of the database says that in the majority of the cases the relationship between attributes is much more difficult to deal with. As an example, the lower part of the Figure 2 plots the attributes 1 and 6: in this case instances belonging to different classes are really mixed, and any classification tool based on centroids might have serious problems in correctly classifying instances.

4.2 The Experiments

As concerns the other classification techniques used for the comparison we have made reference to the Waikato Environment for Knowledge Analysis (WEKA) system release 3.4 [14] which contains a large number of such techniques, divided into groups (bayesian, function–based, lazy, meta–techniques, tree-based, rule-based, other) on the basis of the underlying working mechanism. From each such group we have chosen some representatives. They are: among the bayesian the Bayes Net [15], among the function-based the MultiLayer Perceptron Artificial Neural Network (MLP) [16], among the lazy IB1 [17] and KStar [18], among

Table 1. Image database attributes and their features

attribute	*type*	*minimum*	*maximum*
region centroid column	integer	1	252
region centroid row	integer	11	250
region pixel count	integer	9	9
short line density 5	real	0.000	0.111
short line density 2	real	0.000	0.222
vertical edge mean	real	0.000	25.500
vertical edge standard deviation	real	0.000	572.996
horizontal edge mean	real	0.000	44.722
horizontal edge standard deviation	real	0.000	1386.329
intensity mean	real	0.000	143.444
raw red mean	real	0.000	136.888
raw blue mean	real	0.000	150.888
raw green mean	real	0.000	142.555
excess red mean	real	-48.222	5.777
excess blue mean	real	-9.666	78.777
excess green mean	real	-30.555	21.888
value mean	real	0.000	150.888
saturation mean	real	0.000	1.000
hue mean	real	-2.530	2.864

the meta-techniques the Bagging [19], among the tree-based ones J48 [20] and Naive Bayes Tree (NBTree) [21], among the rule-based ones PART [22] and Ripple Down Rule (Ridor) [23] and among the other the Voting Feature Interval (VFI) [24].

Parameter values used for any technique are those set as default in WEKA.

On the basis of a preliminary tuning phase carried out on this and other databases, DE–C parameters have been chosen as follows: $n = 200$, $g = 500$, $CR = 0.01$ and $F = 0.01$. It is interesting to note that the values for CR and F are much lower than the ones classically used according to literature, which range higher than 0.5. A hypothesis about the reason for this may be that any chromosome in the population consists in this case of $N \cdot C = 19 \cdot 7 = 133$ components, so search space is very large and high values for CR and F would change too many alleles and create individuals too different from the parents. This would drive the search far from the promising regions already encountered, thus creating worse individuals than those present in the current population. As a consequence, given the elitist kind of replacement strategy, only very few new individuals would be able to enter the next generation. The just hypothesized scenario seems confirmed by the evolution shown by the system for high values of CR and F: in this case best individual fitness decrease is not continuous, rather it takes place in steps, each of which lasts tens of generations. This kind of evolution appears to be similar to that typical of a random search with elitism.

Results of DE–C technique are averaged over 20 runs differing one another for the different starting seed provided in input to the random number generator

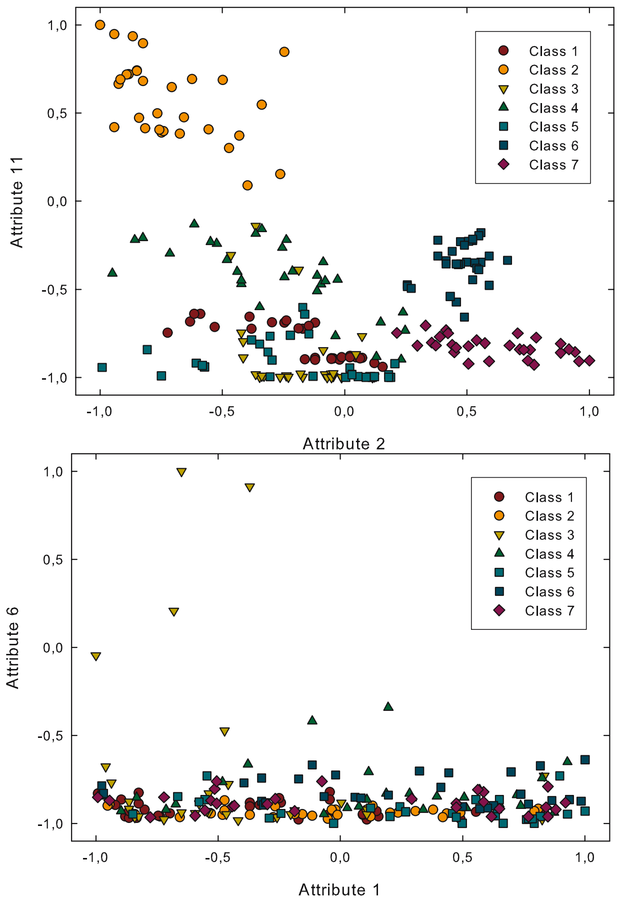


Fig. 2. Some examples of analysis of Image database features

only. For the other techniques, instead, some (MLP, Bagging, Ridor, PART and J48) are based on a starting seed so that also for them 20 runs have been carried out by varying this value. Other techniques (Bayes Net, KStar, VFI) do not depend on any starting seed, so 20 runs have been executed as a function of a parameter typical of the technique (*alpha* for Bayes Net, *globalBlend* for KStar and *bias* for VFI). NBTree and IB1, finally, depend neither on an initial seed nor on any parameter, so only one run has been performed for them.

DE–C execution time is around 27 seconds per run on a personal computer with a 1.6–GHz Centrino processor. Thus times are comparable with those of the other techniques, which range from $2 - 3$ seconds up to about 1 minute for the MLP.

Table 2 shows the results achieved by the 11 techniques on the database. Namely, for any technique the average values of $\%err$ and the related standard deviations σ are given. Of course σ is meaningless for NBTree and IB1.

Table 2. Achieved results in terms of $\%err$ and σ

	DE–C	BAYES NET	MLP ANN	IB1	KSTAR	BAG-GING	J48	NB TREE	PART	RIDOR	VFI
$\%err$	7.46	12.85	7.57	10.71	7.42	10.35	10.71	11.42	10.71	9.64	21.21
σ	0.97	1.09	1.05	—	1.51	2.17	0.00	—	0.00	2.35	0.33

As it can be observed from the values in Table 2 DE–C is the second best technique in terms of $\%err$, very close to KStar which is the best, and closely followed by MLP. All other techniques are quite far from these three. The standard deviation σ for DE–C is not too high, meaning that, independently of the different initial populations, the final classifications achieved have similar correctness. Some techniques like Bagging and Ridor, instead, show very different final values of $\%err$, thus sensitivity to different initial conditions. On the contrary, other techniques like J48 and PART are able to achieve the same final value of $\%err$ in all the effected runs.

Thus, our idea of exploiting DE to find positions of centroids has proven effective to face the Image database.

From an evolutionary point of view, in Fig. 3 we report the behavior of a typical run (the execution shown is the run number 1 carried out on the database).

Its top part shows the evolution in terms of best individual fitness and average fitness in the population as a function of the number of generations. DE–C shows a first phase of about 125 generations in which fitness decrease is strong and almost linear, starting from 0.82 for the best and 0.92 for the average, and reaching about 0.46 for the best and 0.58 for the average. A second phase follows, lasting until about generation 350, in which decrease is slower, and the two values tend to become closer, until they reach 0.23 and 0.26 respectively. From now on the decrease in fitness is slower and slower, about linear again but with a much lower slope, and those two values become more and more similar. Finally, at generation 500 the two values are 0.201 and 0.208 respectively.

The bottom part of Fig. 3, instead, reports the behavior of $\%err$ as a function of the generations. Namely, for any generation its average value, the lowest error value in the generation $\%err_{be}$ and the value of the individual with the best fitness value $\%err_{bf}$ are reported. This figure shows that actually the percentage of classification error on the testing set decreases as distance–based fitness values diminish, thus confirming the hypothesis underlying our approach. It should be remarked here that $\%err_{bf}$ does not, in general, coincide with $\%err_{be}$, and is usually greater than this latter. This is due to the fact that our fitness does not take $\%err$ into account, so evolution is blind with respect to it, as it should be, and it does not know which individual has the best performance on the testing set.

As described above for a specific run is actually true for all the runs carried out, and is probably a consequence of the good parameter setting chosen. This choice on the one hand allows a fast decrease in the first part of the run and, on the other hand, avoids the evolution being stuck in premature convergence as long as generations go by, as it is evidenced by the fact that best and aver-

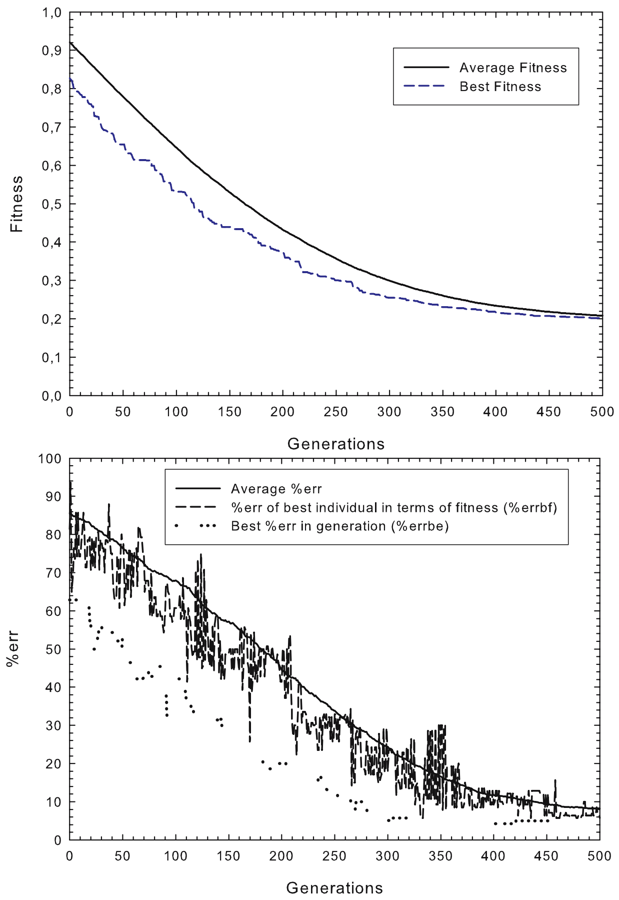


Fig. 3. Typical behavior of fitness (top) and $\%err$ (bottom) as a function of the number of generations

age fitness values are different enough during the whole evolution. Preliminary experiments not reported here showed, instead, that using high values for CR and F leads to best and average fitness values to rapidly become very close.

5 Conclusions and Future Works

The problem of automatically classifying handsegmented parts of images has been faced in this paper by use of Differential Evolution. A publicly available database has been taken into account. A tool has been designed and implemented in which DE is used to find the positions of the class centroids in the search space such that for any class the average distance of instances belonging to that class from the relative class centroid is minimized. The classification performance is estimated by computing the error percentage on the testing set for the resulting best individual. The experimental results have proven that the tool is successful in tackling the task and is very competitive in terms of error percentage on the testing set when compared with other ten classification tools widely used in literature. In fact, only KStar has shown slightly better performance. Execution times are of the same order of magnitude as those of the ten techniques used.

Results seem to imply that our method based on the simple concept of centroid can be fruitfully exploited in the Image database. It may be hypothesized that DE coupled with centroids might be suitably used in general to face classification of instances in databases. This shall be one of our future issues of investigation.

Future works will aim to shed light on the effectiveness of our system in this field, and on its limitations as well. To this aim, we plan to endow DE with niching, aiming to investigate whether this helps in further improving performance.

References

1. Price K, Storn R (1997) Differential evolution. Dr.Dobb's Journal 22(4):18–24.
2. Storn R, Price K (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4):341-359. Kluwer Academic Publishers.
3. Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford Univ. Press.
4. Eiben A E, Smith J E (2003) Introduction to evolutionary computing. Springer.
5. Fayyad U M, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (1996) Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press.
6. Duda R O, Hart P E, Stork D G (2001) Pattern Classification. Wiley-Interscience.
7. Weiss S M, Kulikowski C A (1991) Computer Systems that Learn, Classification and Prediciton Methods from Statistics, Neural Networks, Machine Learning and Expert Systems. Morgan Kaufmann, San Mateo, CA.
8. Gomez-Skarmeta A F, Valdes M, Jimenez F, Marin-Blazquez J G (2001) Approximative fuzzy rules approaches for classification with hybrid–GA techniques. Information Sciences 136(1-4):193–214.
9. Luukka P, Sampo J (2004) Weighted Similarity Classifier Using Differential Evolution and Genetic Algorithm in Weight Optimization. Journal of Advanced Computational Intelligence and Intelligent Informatics 8(6):591–598.
10. Paterlini S, Krink T (2004) High Performance Clustering with Differential Evolution. In: Proceedings of the Sixth Congress on Evolutionary Computation (CEC-2004), vol. 2, pp. 2004-2011. IEEE Press, Piscataway NJ.
11. Omran M, Engelbrecht A P, Salman A (2005) Differential Evolution Methods for Unsupervised Image Classification. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2005). IEEE Press, Piscataway NJ.
12. Blake C L, Merz C J (1998) UCI repository of machine learning databases, University of California, Irvine. http://www.ics.uci.edu/~mlearn/MLRepository.html.
13. Piater J H, Riseman E M, Utgoff P E (1999) Interactively training pixel classifiers. International Journal of Pattern Recognition and Artificial Intelligence 13(2):171–194. World Scientific.
14. Witten I H, Frank E (2000) Data mining: practical machine learning tool and technique with Java implementation. Morgan Kaufmann, San Francisco.
15. Jensen F (1996) An introduction to bayesian networks. UCL Press and Springer-Verlag.

16. Rumelhart D E, Hinton G E, Williams R J (1986) Learning representation by back-propagation errors. Nature 323:533–536.
17. Aha D, Kibler D (1991) Instance-based learning algorithms. Machine Learning 6:37–66.
18. Cleary J G, Trigg L E (1995) K*: an instance-based learner using an entropic distance measure. In: Proceedings of the 12th International Conference on Machine Learning, pp. 108–114.
19. Breiman L (1996) Bagging predictors. Machine Learning 24(2):123–140.
20. Quinlan R (1993) C4.5: Programs for machine learning. Morgan Kaufmann Publishers, San Mateo, CA.
21. Kohavi R (1996) Scaling up the accuracy of naive-bayes classifiers: a decision tree hybrid. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 202–207, AAAI Press, Menlo Park, CA.
22. Frank E, Witten I H (1998) Generating accurate rule sets without global optimization. In: Machine Learning: Proceedings of the Fifteenth International Conference, pp. 144–151. Morgan Kaufmann Publishers, San Francisco, CA.
23. Compton P, Jansen R (1988) Knowledge in context: a strategy for expert system maintenance. In: Proceedings of AI'88, pp. 292–306. Springer-Verlag, Berlin.
24. Demiroz G, Guvenir A (1997) Classification by voting feature intervals. In: Proceedings of the 9th European Conference on Machine Learning.

Mixed-Integer Evolution Strategies and Their Application to Intravascular Ultrasound Image Analysis

Rui Li[1], Michael T.M. Emmerich[1], Ernst G.P. Bovenkamp[2],
Jeroen Eggermont[2], Thomas Bäck[1], Jouke Dijkstra[2], and Johan H.C. Reiber[2]

[1] Natural Computing Group, Leiden University,
P.O. Box 9500, Leiden 2300 CA, The Netherlands
{ruili, emmerich, baeck}@liacs.nl
[2] Division of Image Processing, Department of Radiology C2S,
Leiden University Medical Center,
P.O. Box 9600, Leiden 2300 RC, The Netherlands
{E.G.P.Bovenkamp, J.Eggermont, J.Dijkstra, J.H.C.Reiber}@lumc.nl

Abstract. This paper discusses Mixed-Integer Evolution Strategies and their application to an automatic image analysis system for IntraVascular UltraSound (IVUS) images. Mixed-Integer Evolution Strategies can optimize different types of decision variables, including continuous, nominal discrete, and ordinal discrete values. The algorithm is first applied to a set of test problems with scalable ruggedness and dimensionality. The algorithm is then applied to the optimization of an IVUS image analysis system. The performance of this system depends on a large number of parameters that – so far – need to be chosen manually by a human expert. It will be shown that a mixed-integer evolution strategy algorithm can significantly improve these parameters compared to the manual settings by the human expert.

1 Introduction

Many optimization problems are in practice difficult to solve with standard numerical methods (like gradient-based strategies), as they incorporate different types of discrete parameters, and confront the algorithms with a complex geometry (rugged surfaces, discontinuities). Moreover, the high dimensionality of these problems makes it almost impossible to find optimal settings through manual experimentation. Therefore, robust automatic optimization strategies are needed to tackle such problems.

In this paper we present an algorithm, namely the Mixed-Evolution Strategy (MI-ES), that can handle difficult mixed-integer parameter optimization problems. In particular, we aim to optimize feature detectors in the field of Intravascular Ultrasound (IVUS) image analysis.

IVUS images show the inside of coronary or other arteries and are acquired with an ultrasound catheter positioned inside the vessel. An example of an IVUS image with several detected features can be seen in Figure 1. IVUS images are difficult to interpret which causes manual segmentation to be highly sensitive to intra- and inter-observer variability[5]. In addition, manual segmentation of the large number of IVUS images per patient is very time consuming. Therefore an automatic system is needed. However, feature detectors consist of large numbers of parameters that are hard to optimize manually and may differ for different interpretation contexts. Moreover, these parameters are subject to change when something changes in the image acquisition process.

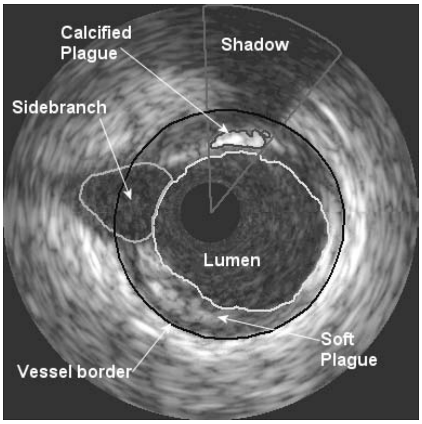


Fig. 1. An IntraVascular UltraSound (IVUS) image with detected features. The black circle in the middle is where the ultrasound imaging device (catheter) was located. The dark area surrounding the catheter is called the *lumen*, which is the part of the artery where the blood flows. Above the catheter *calcified plague* is detected which blocks the ultrasound signal causing a dark *shadow*. Between the inside border of the vessel and the lumen there is some soft plague, which does not block the ultrasound signal. The dark area left of the catheter is a sidebranch.

To optimize these IVUS feature detectors, we propose to use mixed-integer evolution strategies, which have so far been successfully applied to simulator-based optimization of chemical engineering plants [2]. Before applying these strategies on the application problem, their performance and reliability will first be tried on a set of test functions, including a new integer test function with scalable degree of ruggedness.

This paper is structured as follows: first, in Section 2, we will explain the application problem in more detail. Then, in Section 3, mixed-integer evolution strategies (MI-ES) will be introduced. These strategies are tried on artificial test problems in Section 4. Promising variants of the MI-ES are then applied to the parameter optimization of an IVUS feature detector in Section 5. Finally, conclusions and future work are presented in Section 6.

2 Intravascular Ultrasound Image Analysis

IntraVascular UltraSound (IVUS) is a technique used to get real-time high resolution tomographic images from the inside of coronary vessels and other arteries. To gain insight into the status of an arterial segment a so-called catheter pullback sequence is carried out. A catheter (⌀ ±1mm) with a miniaturized ultrasound transducer at the tip is inserted into a patient's artery and positioned downstream of the segment of interest. The catheter is then pulled back in a controlled manner, using motorized pullback (1 mm/s), during which images are acquired continuously.

In [1] a state-of-the-art multi-agent system is used to detect lumen, vessel, shadows, sidebranches and calcified plagues in IVUS images. The system, as shown in Figure 2, is based on the cognitive architecture Soar (States, operators and results) [6]. IVUS image processing agents interact with other agents through communication, act on the world by controlling and adapting image processing operations and perceive that same world by accessing image processing results.

Agents thereby dynamically adapt the parameters of low-level image segmentation algorithms based on knowledge of global constraints, contextual knowledge, local image information and personal beliefs. The lumen-agent, for example, encodes and controls an image processing pipeline which includes binary morphological operations, an ellipse-fitter and a dynamic programming module, and it determines all relevant parameters. Generally, agent control allows the underlying segmentation algorithms to be simpler and to be applied to a wider range of problems with a higher reliability.

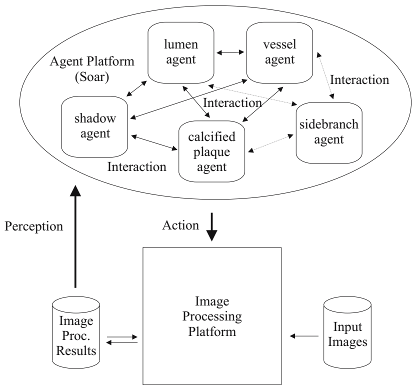


Fig. 2. Global view of the multi-agent system architecture

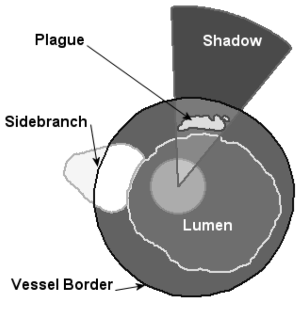


Fig. 3. Schematic version of Figure 1 as detected by the multi-agent image segmentation system

Although the multi-agent system has shown to offer lumen and vessel detection comparable to human experts [1], it is designed for symbolic reasoning, not numerical optimization. Further it is almost impossible for a human expert to completely specify how an agent should adjust its feature detection parameters in each and every possible interpretation context. As a result an agent has only control knowledge for a limited number of contexts and a limited set of feature detector parameters.

In addition, this knowledge has to be updated whenever something changes in the image acquisition pipeline. Therefore, it would be much better if such knowledge might be acquired by learning the optimal parameters for different interpretation contexts automatically.

3 Mixed-Integer Evolution Strategies

From the point of view of numerical analysis, the problem described above can be classified as a black box parameter optimization problem. The evaluation software is represented as an objective function $f : \mathbb{S} \rightarrow \mathbb{R}$ to be minimized, whereby $\mathbb{S}$ defines a parametric search space from which the decision variables can be drawn. Typically, also constraint functions are defined, the value of which has to be kept within a feasible domain. Typically, the objective function evaluation and also partly the evaluation of the constraint function evaluation are done by the evaluation software, which, from the point of view of the algorithm, serves as a black box evaluator.

One of the main reasons why standard approaches for black box optimization cannot be applied to the application problem, is because different types of discrete and continuous decision variables are involved in the optimization. For the parametrization of the image analysis system three main classes of decision variables are identified:

- **Continuous variables:** These are variables that can change gradually in arbitrarily small steps
- **Ordinal discrete variables:** These are variables that can be changed gradually but there are smallest steps (e.g. discretized levels, integer quantities)
- **Nominal discrete variables:** These are discrete parameters with no reasonable ordering (e.g. discrete choices from an unordered list/set of alternatives, binary decisions).

Taking the above into account, the optimization task reads:

$$f(r_1, \dots, r_{n_r}, z_1, \dots, z_{n_z}, d_1, \dots, d_{n_d}) \rightarrow min \tag{1}$$

$$\text{subject to:}$$

$$r_i \in [r_i^{min}, r_i^{max}] \subset \mathbb{R},\ i = 1, \dots, n_r$$

$$z_i \in [z_i^{min}, z_i^{max}] \subset \mathbb{Z},\ i = 1, \dots, n_z$$

$$d_i \in D_i = \{d_{i,1}, \dots, d_{i,|D_i|}\}, i = 1, \dots, n_d$$

Here r_i denotes a continuous variable, z_i a integer variable, and d_i a discrete variable which can be drawn from a predescribed set D_i.

3.1 Evolution Strategies

Mixed-integer evolution strategies (MI-ES) are a special instantiation of evolution strategies that can deal with parameter types, i.e. can tackle problems as described above. They were first proposed in Emmerich et al. [2, 3] for the purpose of chemical engineering plant optimization with process simulators from industry. There, a detailed outline of the algorithm was given with experiments on test functions of higher dimension. However, the experimental analysis of this algorithm deserves some further attention. Therefore, before presenting results of new studies below, we will first give a brief outline of the algorithm and discuss its working principles.

The main loop of the MI-ES is displayed in algorithm 1.. First, the algorithm initializes μ individuals randomly (uniformly distributed within the parameter ranges), and evaluates them by means of the objective function f. Then, λ offspring individuals are created. For each new variant, two individuals are drawn out of the current population $P(t)$ and discrete recombination (= uniform crossover) on the objective variables and intermediate recombination (averaging) on the step-size parameters is used to generate an offspring. Each of the λ offspring generated by this process is then modified by means of the mutation operator, which we will describe in more detail in section 3.2. Then the objective function value of the λ new individuals (= offspring individuals) is evaluated. Next, the selection operator chooses the μ best individuals out of the λ offspring individuals and the μ parental individuals.

Note, that an alternative strategy would be to consider only the offspring population for the selection of individuals in $P(t)$. This strategy would be termed a (μ, λ)-ES. In this paper we choose a Plus-Strategy based on preliminary experiments that indicated that this strategy behaves superior whenever only a small number of experiments could be afforded. As long as the termination criterion[1] is not fulfilled, these μ selected individuals form the next population $P(t+1)$. The proposed algorithmic scheme will be called a $(\mu + \lambda)$-ES.

Algorithm 1. Main loop of a $(\mu + \lambda)$- ES.

```
$t \leftarrow 0$
**initialize** population $P(t)$ of $\mu$ individuals randomly within the individual space $\mathbb{I}$
**evaluate** the $\mu$ initial individuals applying fitness function $f$
**while** Termination criteria not fulfilled **do**
    **recombine** $\lambda$ offspring individuals out of $\mu$ parents, by choosing randomly two
     individuals and recombine them, to obtain each of the offspring individual.
    **mutate** the $\lambda$ offspring individuals
    **evaluate** the $\lambda$ offspring individuals
    **select** the $\mu$ best individuals for $P(t+1)$ from $\lambda$ offspring individuals and $\mu$ parents
    $t \leftarrow t + 1$
**end while**
```

[1] In most cases a maximal number of generations is taken as termination criterion.

In order to allow for a mutative step-size adaptation, a surplus of offspring individuals needs to be generated in each generation. The recommended ratio of $\mu/\lambda \simeq 1/7$ leads to a good average performance of evolution strategies in many cases [8].

3.2 Mutation of Different Variable Types

Individuals consist of three vectors of decision variables and associated to each of these vectors, we also maintain a vector of step-size parameters. In [2] it was proposed to learn only a single step-size for each variable vector. Accordingly, in this case the vector reduces to a scalar value associated with a class of parameters. In summary, a variable vector that represents an individual reads

$$(r_1, \ldots, r_{n_r}, z_1, \ldots, z_{n_z}, d_1, \ldots, d_{n_d}, s_1, \ldots, s_{n_s}, q_1, \ldots, q_{n_\varsigma}, p_1, \ldots, p_{n_p}) \in \mathbb{I},$$

$$\mathbb{I} = \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z} \times D_1 \times \cdots \times D_{n_d} \times (\mathbb{R}^+)^{n_s} \times (\mathbb{R}^+)^{n_q} \times [0,1]^{n_p}.$$

In case $n_s = n_r$ the step-sizes are assigned to the different vector positions and can be adapted individually. The step-size parameters for the continuous variables s_i represent standard-deviations used for scaling the mutation distribution of the object variables. In case $n_s = 1$ the same step-size is used for all parameters. The step-sizes for the integer variables are defined in a similar manner. As we will discuss later, another distribution will be used to sample integer values. Finally, the step-size parameters for the nominal discrete variables are interpreted as mutation probabilities.

Algorithm 2. Mutation procedure in MI-ES

```
for $i = 1, \ldots, n_r$ do
  $s'_i \leftarrow s_i \exp(\tau_g \mathrm{N}_g + \tau_l \mathrm{N}(0,1))$
  $r'_i = r_i + N(0, s'_i)$
end for
for $i = 1, \ldots, n_z$ do
  $q'_i \leftarrow q_i \exp(\tau_g \mathrm{N}_g + \tau_l \mathrm{N}(0,1))$
  $z'_i \leftarrow z_i + \mathrm{G}(0, q'_i)$
end for
$p'_i := 1/[1 + \frac{1-p_i}{p_i} * \exp(-\tau_l * \mathrm{N}(0,1))]$
for $i \in \{1, \ldots, n_d\}$ do
  if $\mathrm{U}(0,1) < p'_i$ then
    $d'_i \leftarrow$ uniformly randomly value from $D_i$
  end if
end for
```

Algorithm 2 summarizes the mutation procedure. For the local and global step-size learning rates τ_l and τ_g we use the recommended parameter settings $\tau = 1$ and $\tau_l = 1/\sqrt{2\sqrt{n_r}}$ and $\tau_g = 1/\sqrt{2n_r}$ (cf. [8]).

Note, that in case a single step-size is chosen, $\tau_l = 0$. In the algorithm $\mathrm{N}(0,1)$ denotes a function that results in a standard normal distributed random number. Accordingly, $\mathrm{U}(0,1)$ denotes a function returning a uniformly distributed number in $[0,1] \subset \mathbb{R}$ and $\mathrm{G}(0,q)$ returns a geometrically distributed random value.

Among these distributions, the geometric distribution deserves further attention, as it is rarely referred to in literature. Rudolph [7] proposed the geometrical distribution on integer representations. Geometrical distributed random variables are random variables whose values are in $\mathbb{Z}$. They have properties similar to normal distributed random variables in $\mathbb{R}$. In particular, they have infinite support in $\mathbb{Z}$, are unimodal with a peak in 0, and their probability function is symmetric in the origin. Moreover, as pointed out by Rudolph [7], multivariate extensions are characterized by a rotational symmetry with regard to the ℓ_1 norm[2] and they belong to a family of maximal entropy distributions. Finally, by increasing the value q the standard deviation of the random variable can be gradually increased. All these characteristics make the geometric distribution well suited for application within mixed-integer evolution strategies.

A geometrically distributed random variable G can be generated from two uniformly distributed random variables $u_1 := U(0,1); u_2 := U(0,1)$ via:

$$\mathrm{G} = G_1 - G_2,\ p = 1 - \frac{s/n_z}{1+\sqrt{1+(\frac{s}{n_z})^2}},\ G_i = \left\lfloor \frac{\ln(1-u_i)}{\ln(1-p)} \right\rfloor, i = 1,2 \qquad (2)$$

The mutation of the mutation probabilities is done by means of a logistic distribution as described in [2]. To make sure that variables stay within their respective boundaryes we have added some routines for interval treatment to the MI-ES. While for the continuous variables we used reflection at the boundary, for the integer variables we set the value to the bound, whenever the bound is exceeded. The latter method is also used to keep the mutation probabilities within bounds.

4 Study on Artificial Test Problems

In order to select a favorable variant of the MI-ES for the time-consuming runs on the image analysis problem, we study the behavior of the MI-ES on a generalized sphere function and barrier problems using a new problem generator.

The generalized sphere function is an extension of a standard problem [2]:

$$f_{sphere}(\mathbf{r},\mathbf{z},\mathbf{d}) = \sum_{i=1}^{n_r} r_i^2 + \sum_{i=1}^{n_z} z_i^2 + \sum_{i=1}^{n_d} d_i^2 \rightarrow \min \qquad (3)$$

$$n_r = n_z = n_d = 7, \mathbf{r} \in [-10,10]^{n_r} \subset \mathbb{R}^{n_r}, \mathbf{z} \in [-10,10]^{n_z}, \mathbf{d} \in \{0,\dots,5\}^{n_d} \quad (4)$$

This problem is relatively simple, as it is decomposable and unimodal. We use it to gain some insights of how the MI-ES behaves on rather simple problems and thus to estimate the best case behavior of the MI-ES.

[2] Sum of absolute values.

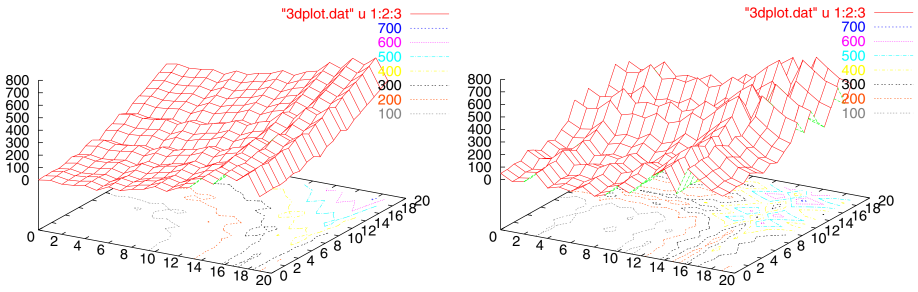


Fig. 4. Surface plots of the barrier function for two variables. All other variableswere kept constant at a value of zero, two integer values were varied in the range from 0 to 20.

As a more complex test case we designed the multimodal barrier problem. It produces integer optimization problems with a scalable degree of ruggedness (determined by parameter C) by generating an integer array `A` using the following algorithm:

```
$\mathtt{A}[i] = i, i = 0, \ldots, 20$
**for** $k \in \{1, \ldots, C\}$ **do**
    $j \leftarrow$ random number out of $\{0, \ldots, 19\}$
    swap values of $\mathtt{A}[j]$ and $\mathtt{A}[j+1]$
**end for**
```

Then a barrier function is computed:

$$f_{barrier}(\mathbf{r}, \mathbf{z}, \mathbf{d}) = f_{sphere}(\mathbf{r}, \mathbf{d}) + \sum_{i=1}^{n_z} \mathtt{A}[z_i]^2 \rightarrow \min \tag{5}$$

$$n_r = n_z = n_d = 7, \mathbf{r} \in [-10, 10]^{n_r} \subset \mathbb{R}, \mathbf{z} \in [0, 20]^{n_z}, \mathbf{d} \in \{0, \ldots, 5\}^{n_d} \tag{6}$$

The parameter C controls the ruggedness of the resulting function with regard to the integer space. High values of C result in rugged landscapes with many barriers. To get an intuition about the influence of C on the geometry of the function we included plots for a two-variable instantiation of the barrier function in Figure 4 for $C = 20$ and $C = 100$.

In Figure 5 we present some parameter studies of the MI-ES. We compare different settings for the population and offspring sizes $\{(\mu = 3, \lambda = 10), (\mu = 4, \lambda = 28), (\mu = 15, \lambda = 100)\}$ on the sphere and barrier problems. It turns out that the $(\mu = 4, \lambda = 28)$ setting performs best on the more difficult problems, at least if only a limited number of about 2000 function evaluation can be afforded (as in our application problem). However, the plots on the barrier problem show that for long runs with $t \gg 2000$ a strategy with a larger population size might be favorable. We also observed that it is less risky with regard to the performance of the strategy to use a single step-size per parameter type ($n_s = n_q = n_p = 1$), instead of individual step-sizes for all of the variables. This corresponds to the findings in [2].

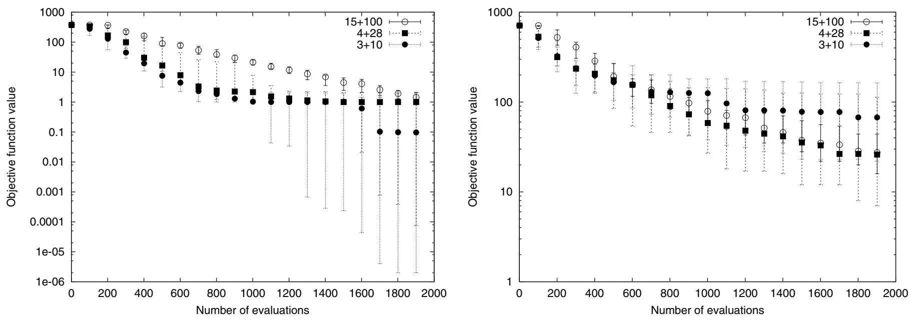


Fig. 5. Averaged results on the sphere and the barrier functions. For all results the median and the quartiles of 20 runs are displayed.

5 Experimental Results on Image Analysis Problem

To determine whether or not MI-ES can be used as an optimizer for the parameters of feature detectors in the multi-agent image analysis system we test MI-ES on the lumen feature detector. This detector is chosen, because it can produce good results in isolation without additional information about sidebranches, shadows, plagues and vessels.

Table 1 contains the parameters for the lumen feature detector together with their type, range, dependencies and the default settings determined by an expert. As can be seen the parameters are a mix of continuous, nominal discrete (integer) and ordinal discrete(including boolean) variables.

For the experiments we use five disjoint sets of 40 images. The fitness function used in the experiments is based on the difference (see Eq. 7)between the contour c found by the lumen feature detector and the desired lumen contour C drawn by a human expert. The difference measure is defined as the sum of the distances of the points of contour c that are more than a *threshold* distance away from contour C. The reason to allow for a small difference between the two contours is that even an expert will not draw the exact same contour twice in a single IVUS image. The fitness function itself is the calculated average difference over the 40 images in the dataset.

Let *#points* denote the total number of points of contour C, then the contour difference is defined as:

$$\mathit{difference}(c, C) = \sum_{p=1}^{\#points} d(c_p, C), \ \ \mathit{if} \ \ d(c_p, C) > \mathit{threshold} \tag{7}$$

On each of the 5 datasets we trained our (4 + 28) MI-ES algorithm for 100 iterations resulting in 2804 fitness evaluations. The results are displayed in Table 2. Parameter solution 1 was trained on dataset 1, parameter solution 2 was trained on dataset 2, etc ....

Table 2 shows that for most cases the MI-ES parameter solutions result in lower average contour differences when applied to both test- and training data.

Table 1. Parameters for the lumen feature detector

name	type	range	dependencies	default
	—			
maxgray	integer	[2, 150]	> mingray	35
mingray	integer	[1, 149]	< maxgray	1
connectivity	ordinal	4,6,8		6
relativeopenings	boolean	{false,true}		true
nrofcloses	integer	[0, 100]	used if not relativeopenings	5
nrofopenings	integer	[0, 100]	used if not relativeopenings	45
scanlinedir	ordinal	{0,1,2}		1
scanindexleft	integer	[-100, 100]	< scanindexright	-55
scanindexright	integer	[-100, 100]	> scanindexleft	7
centermethod	ordinal	{0,1}		1
fitmodel	ordinal	{ellipse, circel}		ellipse
sigma	continuous	[0.5 10.0]		0.8
scantype	ordinal	{0,1,2}		0
sidestep	integer	[0, 20]		3
sidecost	continuous	[0.0, 100]		5
nroflines	integer	[32, 256]		128

Table 2. Performance of the best found MI-ES parameter solutions when trained on one of the five datasets (parameter solution *i* was trained on dataset *i*). All parameter solutions and the (default) expert parameters are applied to all datasets. Average difference (fitness) and standard deviation w.r.t. expert drawn contours are given.

	default parameters		parameter solution 1		parameter solution 2		parameter solution 3		parameter solution 4		parameter solution 5	
dataset	fitness	s.d.	fitness	s.d.	fitness	s.d.	fitness	s.d.	fitness	s.d.	fitness	s.d.
1	395.2	86.2	148.4	39.5	159.8	43.5	185.4	43.0	144.8	42.0	271.0	74.8
2	400.2	109.2	183.3	59.2	180.7	58.4	207.2	69.2	232.7	71.0	352.0	73.1
3	344.8	65.6	205.9	69.8	203.9	70.1	164.4	49.7	183.9	80.3	327.1	55.9
4	483.1	110.6	284.4	92.7	269.0	73.2	250.4	80.4	173.2	64.7	330.1	82.2
5	444.2	90.6	368.4	100.9	370.9	102.5	462.2	377.3	168.7	64.0	171.8	54.5

Only parameter solution 3 applied to dataset 5 has a higher average contour difference (444.2 vs 462.2). To determine if the best results obtained by the MI-ES algorithm are also significantly better than the default parameter results, a paired two-tailed t-test was performed on the (40) difference measurements for each image dataset and each solution using a 95% confidence interval (p = 0.05). The t-test shows that all differences are significant except for the difference between parameter solution 3 applied to dataset 5 and the default solution. Therefore we conclude that the MI-ES solutions are significantly better than the default parameter solution in 96% of the cases (24 out of 25) and equal in one case.

Some other interesting results shown in Table 2 are that the solution trained with dataset 4 performs better on dataset 1 than the solution which was optimized with dataset 1 although this difference is not significant. Dataset 4 is interesting anyway, as it is the only dataset for which the performance of all parameter solutions are mutually significantly different, while the solution trained with this dataset has the best average performs on all other datasets. Visual inspection of the results of the application of parameter solution 4 to the other datasets shows that this solution is a good *approximator* of the lumen contours in the other datasets, but that the particular solutions trained with those datasets follow the expert contours more closely. Perhaps dataset 4 contains features of the other datasets (1,2,3 and 5) which may explain this behavior. However, visual inspection of the the image datasets does not show any apparent differences.

6 Conclusions and Outlook

In this paper we have applied Mixed-Integer Evolution Strategies (MI-ES) to optimize the parameters of a lumen feature detector for IntraVascular Ultrasound (IVUS) images. The mixed-integer evolution strategy uses state of the art type of variation operators for mixed integer representations, where the parameters are a combination of continuous, ordinal discrete and nominal discrete variables. Different instantiations of the MI-ES are tested on two artificial test problems in order to identify a favorable parameter setting for the experiments on the application problem. Based on the results of this study, a $(4 + 28)$-MI-ES with a single step-size for each variable class was identified as a robust strategy and used for tackling the image analysis problem. The results show a significant improvement over the parameters tuned manually by an expert.

Our first results clearly indicate the feasibility of the MI-ES approach for optimizing parameters of feature object detectors. However, our tests are at the moment restricted to the lumen feature detector. In future studies we plan to investigate the potential of the method for the other feature detectors needed for IVUS image analysis.

We do not expect to find one optimal solution for each feature detector to work in all possible contexts and for all possible patients. Therefore we are going to apply the methods outlined in this paper to different image interpretation contexts which should result in a set of optimal image feature detector solutions rather than in a single solution. The aim is then to let an agent in the multi-agent system decide which particular solution to use based on its current knowledge of the situation. We also intend to further study the mixed-integer evolution strategy algorithm both in theory and practice to get more insight into its strengths and weaknesses.

Acknowledgements

This research is supported by the Netherlands Organisation for Scientific Research (NWO) and the Technology Foundation STW. We would like to thank Andreas Fischbach and Anne Kunert for the experiments on the sphere and barrier problems and the anonymous reviewers for their useful comments.

References

1. E.G.P. Bovenkamp, J. Dijkstra, J.G. Bosch, and J.H.C. Reiber. Multi-agent segmentation of IVUS images. *Pattern Recognition*, 37(4):647–663, April 2004.
2. M. Emmerich, M. Schütz, B. Gross and M. Grötzner: Mixed-Integer Evolution Strategy for Chemical Plant Optimization. In I. C. Parmee, editor, Evolutionary Design and Manufacture (ACDM 2000), 2000, pp. 55-67, Springer NY
3. M. Emmerich, M. Grötzner and M. Schütz: Design of Graph-based Evolutionary Algorithms: A case study for Chemical Process Networks. Evolutionary Computation, 9(3), 2001, pp. 329 - 354, MIT Press
4. F. Hoffmeister and J. Sprave. Problem independent handling of constraints by use of metric penalty functions. In L. J. Fogel, P. J. Angeline, and Th. Bäck, editors, Evolutionary Programming V - Proc. Fifth Annual Conf. Evolutionary Programming (EP'96), pages 289–294. The MIT Press, 1996.
5. G. Koning, J. Dijkstra, C. von Birgelen, J.C. Tuinenburg, J. Brunette, J-C. Tardif, P.W. Oemrawsingh, C. Sieling, and S. Melsa. Advanced contour detection for three-dimensional intracoronary ultrasound: validation – in vitro and in vivo. *The International Journal of Cardiovascular Imaging*, (18):235–248, 2002.
6. A. Newell. *Unified Theories of Cognition*. Number ISBN 0-674-92101-1. Harvard University Press, Cambridge, Massachusetts, 1990.
7. G. Rudolph, An Evolutionary Algorithm for Integer Programming, In Davidor et al.: Proc. PPSN III, Jerusalem, Springer, Berlin, 1994, 139–148.
8. H.-P. Schwefel: Evolution and Optimum Seeking, Sixth Generation Computing Series, John Wiley, NY, 1995

The Honeybee Search Algorithm for Three-Dimensional Reconstruction

Gustavo Olague and Cesar Puente

Proyecto Evovisión,
Departamento de Ciencias de la Computación, División de Física Aplicada,
Centro de Investigación Científica y de Estudios Superiores de Ensenada,
Km. 107 carretera Tijuana-Ensenada, Ensenada, 22860, B.C., México
{olague, puente}@cicese.mx
http://cienciascomp.cicese.mx/Pagina-Olague.htm

Abstract. This paper investigates the communication system of honeybees with the purpose of obtaining an intelligent approach for three-dimensional reconstruction. A new framework is proposed in which the 3D points communicate between them to achieve an improved sparse reconstruction which could be used reliable in further visual computing tasks. The general ideas that explain the honeybee behavior are translated into a computational algorithm following the evolutionary computing paradigm. Experiments demonstrate the importance of the proposed communication system to reduce dramatically the number of outliers.

1 Introduction

Three-dimensional reconstruction has always been a fundamental research topic in computer vision and photogrammetry. Today the importance of image and vision computing task has gained relevance in the evolutionary computing community. This paper proposes a bioinspired approach to tackle the problem of sparse and quasi-dense reconstruction using as model the honeybee search behavior. This work is also inspired by the work of Louchet [12, 1, 13] in which an individual evolution strategy was applied to obtain a three-dimensional model of the scene using stereo-vision techniques. The main characteristic of that work was the application of the Parisian approach to the evolution of a population of 3D points, called flies, in order to concentrate those points on the object surface of the scene. For more about the Parisian approach we recommend [7] and [2]. One of the drawbacks of the approach of Louchet was the lack of a paradigm to provide those 3D points with intelligent capabilities. Indeed, a high number of outliers were produced with their technique. We decide to explore the honeybee search behavior in order to develop an intelligent algorithmic process. Honeybees are considered to perform one of the most complex communication tasks, in the animal world. Indeed, concepts of memory attention, recognition, understanding, interpretation, agreement, decision-making, and knowledge, as well as questions about cognition and awareness, have appeared regularly in the honeybee literature. In this way, the honeybees are considered to achieve mental tasks like remembering, recognizing, searching, finding, understanding, and even disbelieving. All of these tasks are considered major subjects in computer vision and we believe that an algorithm inspired from the honeybee behavior could provide new insights in old problems not yet solved.

2 The Honeybee Dance Language

Currently, most scientists in the honeybee behavioral community agree that the communication system of the bees is a language regarding insect capacities [3]. The honeybee dance language has been used by researchers to create machine vision systems [19, 20], as well as for robotics tasks [11]. All these works attempt to provide knowledge based on the study of the honeybee. However, none of these works have used the adaptive behavior of the honeybee swarm. In this way, our work is also related to the ant colony optimization meta-heuristic and is more general field called swarm intelligence [5, 6]. However, our work is also strongly related to evolutionary computing as we will explain later. This work is part of our own effort to build new algorithms based on some basic principles taken by the observation of a particular natural phenomenon [16, 18]. Honeybees use a sophisticated communication system that enables them to share information about the location and nature of resources. If a sugar solution is placed outdoors a long time might elapse before they found the food. Soon after this first visit, however, bees soon began swarming around the feeder. The communication among bees is performed using what is called the "dance language" as a means of recruitment. The dance language refers to patterned repetitive movements performed by bees that serve to communicate to their nestmates the location of food sources or nest sites. In this way, the dance is a code that conveys the direction, distance, and desirability of the flower patch, or other resource, discovered. The waggle dance of honeybees can be thought of as a miniaturized reenactment of the flight from the hive to the food or resource. Some honeybee scientists have correlated the distance to the site with the speed of the dance. As the flight to the food distance becomes longer, the duration of the waggle portion of the dance becomes longer. However, the detailed nature of distance communication has been difficult to determine, because the rate of circling and the length of the waggle run correlate with distance information. Moreover, a question arise: if is the finding that it is not distance *per se* the bees indicate, but rather the effort needed to arrive at the dance location. What is really important is that honeybees use the dance's symbolically encoded information to locate resources. Thus, honeybees use both dancing and odors to identify the location of resources, as well as the desirability of a resource. The desirability is expressed in the dance's "liveliness" and "enthusiasm": the richer the source, the livelier the dance that can last many minutes, even hours. The dances are deployed to meet various colony needs such as: changed to monitor shifting environmental conditions, responsive to communication with hivemates, and switched on the basis of superior information from other dancers. Hence, these features suggest that the dance is a tool used by the bees, rather than a behavioral pattern rigidly emitted. When a honeybee discovers a rich patch, she returns and seeks out her hivemates in a specific location near the hive entrance called the "dance floor". She performs the dance on the vertical comb in the dark hive surrounded by numerous potential recruits. The dancer pauses for antennal contact with her followers, and to transfer some of the nectar she has harvest to them. The communicative nature of the dance is apparent in that dances are never performed without audience. While the dance is mostly used to indicate the location of flowers, it is also used for pollen, water when the hive is overheating, waxy materials when the comb needs repair, and the new living quarters when part of the colony must relocate. The angle that a bee flies during the flight to the resource, relative to the sun azimuth (the horizontal component of the direction toward the sun), is mirrored in the angle on the comb at which the waggle portion of the dance is performed. If the resource is to be found directly toward the sun, a bee will dance straight upward. If the resource is directly away from the sun, the bee will dance straight downward. If the resource is at 45° to the right of the sun, then the dance is performed with the waggle run at 45° to the right of the vertical, and so forth. Honeybees make a transition from round dances for food near the nest to waggle dances at a greater distance. In fact the bees perform the round dance as the waggle dance being performed on the same spot first in one direction and then in the other. The bees trace out a figure-of-eight with its two loops more or less closely superimposed upon one another. In this way, the waggle dance is represented at its minimal measure of a single point.

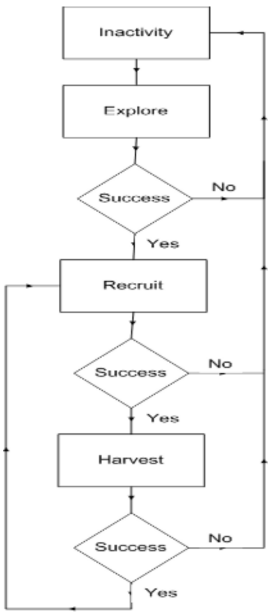


Fig. 1. The honey bee search process is composed of three main activities: exploration, recruitment and harvest

These ideas can be represented as a flow diagram in order to develop an algorithm. Figure 1 shows the flow diagram of the search process employed by the honeybees. The honeybee algorithm that we are proposing is composed by three main activities: exploration, recruitment and harvest. We would like to point that this process is inherently parallel and the algorithm that we are currently using could be further enhanced. The honeybee pass from an inactivity state to the exploration stage in which the "scouts" travel considerable distances to investigate potential sources, and then return and dance to recruit foragers. The sharing of information about the location of sources such as: nectar, pollen, water, and propolis; makes it possible for a honeybee colony to serve as an information center. This communication system allows the reconnaissance of its many foragers, surveying a vast area around the nest, to be used in the discovery of the best sources. Once the exploration is started the recruitment and harvest stages are initialized, and the whole cycle is repeated indefinitely only changed by the current requirement of the hive.

3 The Honeybee Search Algorithm

In this section we give details about the algorithm that we are proposing to obtain information about the three-dimensional world. Normally, the reconstruction of the three-dimensional world is achieved using calibrated and uncalibrated approaches in which several geometric relationships between the scene and the images are computed from point correspondences. The projection matrix models the transformation from the scene to the image, and this could be thought as a direct approach. On the other hand, the transformation from the images to the scene is realized by a process known as triangulation and this could be imagined as an inverse approach. Obviously, to triangulate a 3D point it is necessary to use two 2D points obtained from two images separated at least by a translation. We would like to stay that errors on the calculation could produce misleading results. Therefore it is necessary to apply the best possible algorithm in the calculation of the projection matrix. The problem in this work is posed as a search process in which the 3D points are searched using the direct approach. In this way, it is avoid the use of the epipolar geometry computation. This idea represents a straightforward approach in which a 3D point with coordinates (X, Y, Z) on the Euclidean world is projected into two 2D points with coordinates (x_l, y_l) for the left camera coordinate system and (x_r, y_r) for the right camera coordinate system. A measure of similarity is computed with the *Zero Normalized Cross-Correlation* (ZNCC) and the image gradient to decide if both image points represent the same 3D point. We apply an evolutionary algorithm similar to evolution strategies $(\mu + \lambda)$ in which mutation and crossover are applied as the main search operators.

In this work, we follow the approach proposed by Boumaza in which the new population is created independently by the addition of three different process, see Figure 3. This process is used by the exploration and harvest stages in the honeybee search algorithm, see Figure 2. The exploration stage starts creating

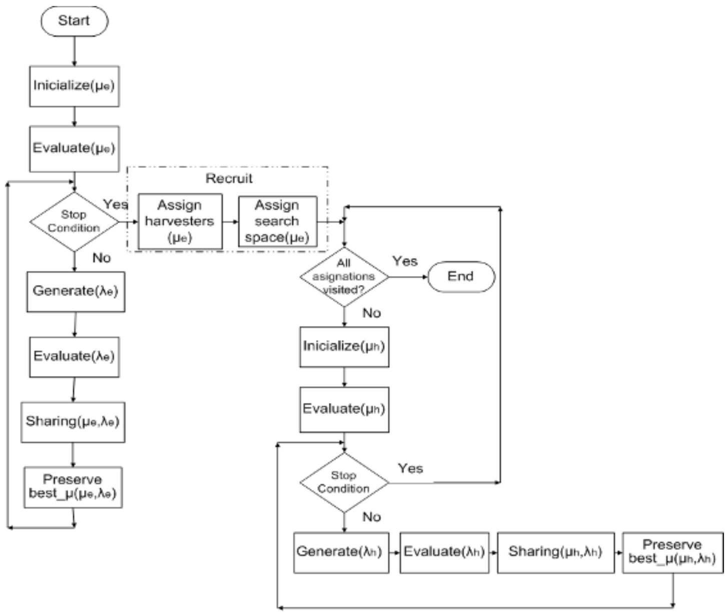


Fig. 2. Flow chart describing the honeybee search algorithm

a random population μ_E of 3D points called explorers, which are then transformed into a new population λ_E using the mutation, crossover and random steps. This stage attempts to simulate the natural process in which the bees explore asynchronously the space in search of the food source. The selection of the best explorers is made with a tournament selection after being evaluate together with the old population. We apply a sharing step in order to balance the distribution of the explorers in the Euclidean world. We repeat this stage until a given number of generations $n = 30$. Then, the recruitment stage is started. Each explorer recruits a number of foragers proportionally to the fitness function. The size of the search space is proportional to the distance between the pair of cameras (hive) and the current 3D point (explorer). Obviously the explorers that are closer to the hive should have a bigger search space, compared with the explorers that are farther away. We start with a fixed size ζ to the nearest visited place near the hive. Then, as long as the bees are farther away from this initial bee; the search space starts to be reduced using as information the distance on the images in order to have an evaluation about the depth in which the points are located.

$$d_i = \sqrt{(x_l - x_r)^2 + (y_l - y_r)^2}.$$

Now, we can proceed to reduce the search space with the following relationship:

$$\begin{array}{c} f = 0.5 \times (1 - u) + 1 \times u \ , \\ \zeta_i' = \zeta_i \times f \ . \end{array} \tag{1}$$

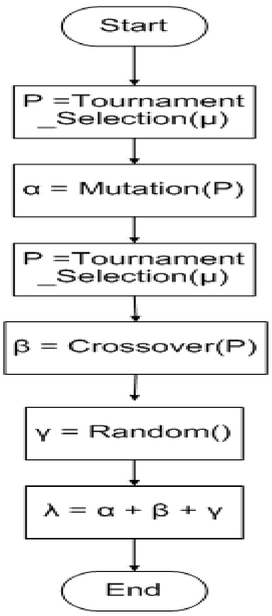


Fig. 3. Flow diagram detailing the generation of a new population

Where u represents the degree of desirability that a place holds according to the distance d_i/d_{max}. The value of f lies in the interval $[0.5, 1]$, where 0.5 is related to the highest distance, while 1 is related to the closest 3D point.

The next stage is to harvest the source patch for each explorer using a similar algorithm with two cycles. The first cycle is dedicated to visit each place that was selected by the explorer. In this way, the foragers that have been selected by the explorer starts a new search process around the point where the explorer is located in order to exploit this location. Hence, the exploration and exploitation steps are achieved by the explorers and foragers respectively. As we can observe each group of foragers exploits sequentially all places. Note that the number of foragers that have been assigned to each explorer is variable according to the fitness function. It is possible that not all explorers have assigned foragers to harvest their place location. In order to know how many foragers are assigned to each explorer, we calculate the proportion of foragers being assigned to the explorers using the proportional fitness

$$p_i = fitness_i / \sum_{j=1}^{N} fitness_j \ .$$

Thus, the number of foragers assigned to each explorer is computed using the following factor

$$r_i = p_i * \lambda \ , \tag{2}$$

where λ is the total size of the population. The second cycle is similar to the exploration stage. Here, the fitness function computation uses besides the ZNCC the homogeneity of the texture without gradient computation . The homogeneity is computed using the *Gray Level Coocurrence Matrix* because it has been proved reliable in image classification and segmentation for content based image retrieval [10]. Also, the size of the search space is obviously smaller with respect to the exploration stage where it is considered the whole space. However, the number of bees could be even bigger with respect to the exploration stage because the total number of foragers is much bigger than the total number of explorers. Here, we use 200 explorers and 2000 foragers. Next we explain the main search operators.

3.1 Evolutionary Search Operators: Mutation, Crossover, and Sharing

The honeybees are recombined coordinate by coordinate using the SBX crossover operator [4]. The SBX operator emulates the working principle of the single point crossover operator on binary strings. From two parent solutions P_1 and P_2, it creates two children C_1 and C_2 as follows:

$$C_1 = 0.5[(1+\beta)P_1 + (1-\beta)P_2]$$
$$C_2 = 0.5[(1-\beta)P_1 + (1+\beta)P_2]$$

$$\text{with } \beta = \begin{cases} (2u)^{\frac{1}{\eta_x+1}} & \text{if } u < 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta_x+1}} & \text{otherwise.} \end{cases}$$

The spread factor β is dependent on a random variable $u \in [0,1]$ and on an user defined nonnegative value η_x that characterizes the distribution of the children in relation to their parents.

Mutation is applied to each of the real variables using a polynomial distribution perturbation. The mutation operation modifies a parent P into a child C using the boundary values $P^{(LOW)}$ and $P^{(UP)}$ of each of the decision variables in the following manner:

$$C = P + (P^{(UP)} - P^{(LOW)})\delta$$

$$\text{with } \delta = \begin{cases} (2u)^{\frac{1}{\eta_m+1}} - 1 & \text{if } u < 0.5 \\ 1 - [2(1-u)]^{\frac{1}{\eta_m+1}} & \text{otherwise .} \end{cases}$$

A novel representation proposed in [14],[15] is used for the real-coded evolutionary operators. This consists in encapsulating both crossover and mutation into a single algebraic affine transformation. Since two real-coded variables Y_1 and Y_2 represent a point in the affine plane, an affine transformation of the form

$$X_1' = b_{11}X_1 + b_{12}X_2 + C_1$$
$$X_2' = b_{21}X_1 + b_{22}X_2 + C_2$$

is applied, where the coefficients are arbitrary real numbers, subject to $|b_{rs}| \neq 0$. This transformation can be extended to include the n variables contained in two different solutions. Accordingly, the generation of new solutions within the evolutionary algorithm can be stated as follows:

$$\begin{pmatrix} X'_{1_1} & Y'_{1_1} & Z'_{1_1} & \ldots & Z'_{1_n} \\ X'_{2_1} & Y'_{2_1} & Z'_{2_1} & \ldots & Z'_{2_n} \end{pmatrix} = \begin{bmatrix} b_{11} & b_{12} & C_1 \\ \underbrace{b_{21} \quad b_{22}}_{Crossover} & & \underbrace{C_2}_{Mutation} \end{bmatrix}_n \begin{pmatrix} X_{1_1} & Y_{1_1} & Z_{1_1} & \ldots & Z_{1_n} \\ X_{2_1} & Y_{2_1} & Z_{2_1} & \ldots & Z_{2_n} \\ 1 & 1 & 1 & \ldots & 1 \end{pmatrix}$$

The advantages of this encapsulation are:

1. Standardized treatment of all transformations
2. Complex transformations are composed from simple transformations by means of matrix multiplication.
3. Simple inversion of the transformation by matrix inversion.
4. Extremely fast, hardware supported matrix operations in high-power graphic workstations.

Finally, we applied a 3D sharing to the honeybees in order to balance the diversity of solutions. In the work of Louchet a 2D sharing was applied with the idea of simplifying the computation. However, this has the drawback of incorrectly penalizing those 3D points that projects into the same image location without being actually around the same 3D space. Thus, we decide to use the sharing proposed by Goldberg and Richardson [9]

$$Sh(d_{i,j}) = \begin{cases} 1 - \frac{d_{(i,j)}}{\sigma_{share}} \text{ , if } & d_{i,j} \leq \sigma_{share} \\ \quad 0 & otherwise \end{cases}$$

where $d_{(i,j)}$ is the distance between the individuals i and j. σ_{share} is the threshold that controls the ratio of sharing. The above function is applied to each individual to obtain a niche count as follows: $n_i = \sum_{j=1}^{N} Sh(d_{i,j})$. Then the shared fitness function is calculated with the following expression $fitness'_i = \frac{fitness_i}{n_i}$.

4 Experimental Results and Conclusions

We have applied the honeybee search algorithm described in this paper on several pair of images. Here for reason of space we show only the results that we have obtained with two stereo pairs. Those images were captured with a Pulnix digital camera TM-9701d with a C-mount Fujinon lens HF16A-2M1, of focal length $f = 16mm$. We describe now the parameters that we have used in each stage of the algorithm. The exploration stage uses a parent population $\mu_E = 200$, and a child population $\lambda_E = 500$. The child population is generated according to the following rates: mutation $\alpha_E = 0.6$, crossover $\beta_E = 0.1$, and random $\gamma_E = 0.3$. The harvest stage uses a parent population of $\mu_H = 2000$ and a child population $\lambda_H = 4000$. The rates are the same of the exploration stage. The parameters of the recruitment stage are automatically computed as we have explained in

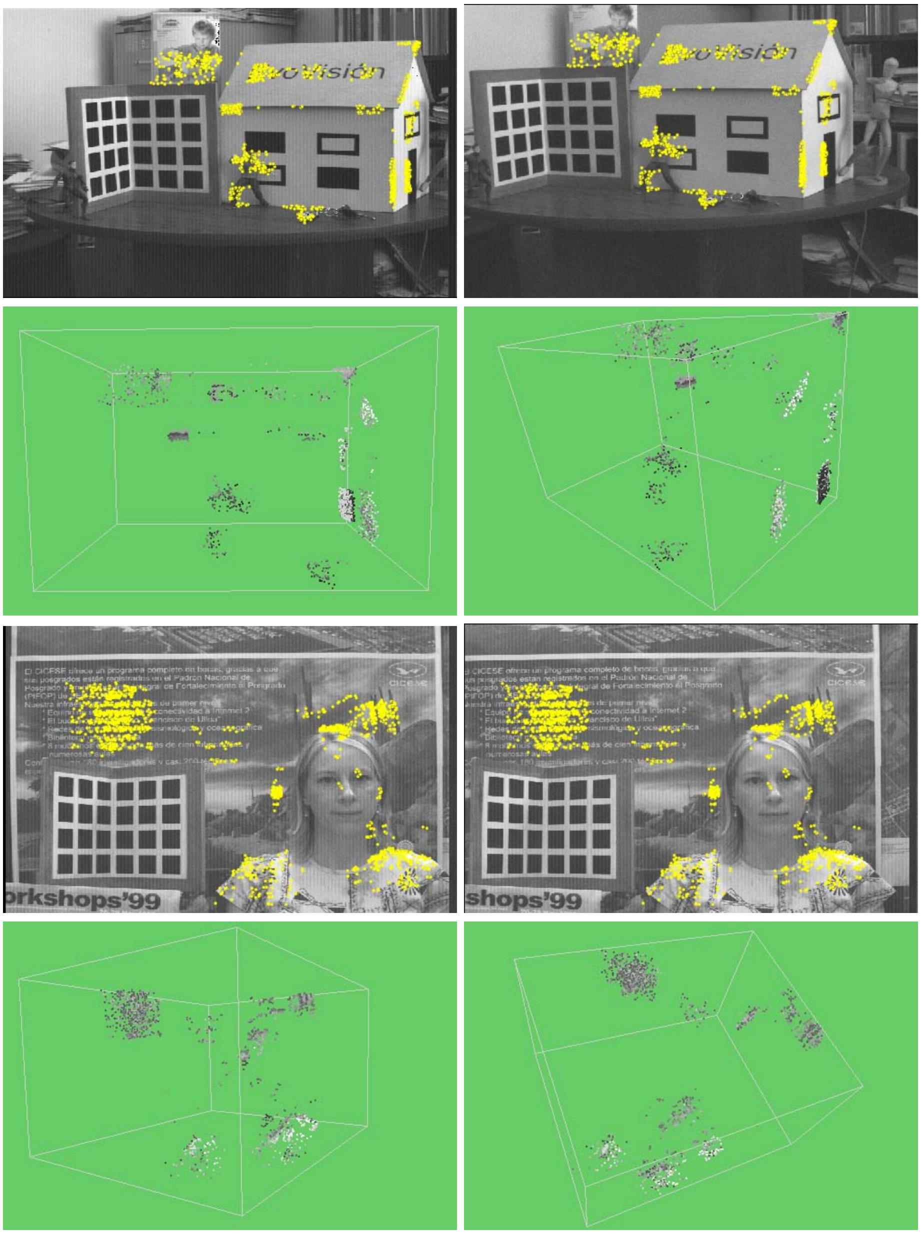


Fig. 4. These figures shows the results of applying the honeybee search algorithm. The first two images shows the first stereo pair with the projection of the artificial honeybees, while the second row shows the *VRML* to appreciate the spatial coherence. The third and fourth rows show the results with a real person.

the document, see Equations 1 and 2. The sharing uses the following parameter $\sigma_{share} = 100mm$. Note that the objects on both images are placed more or less at the same distance to the stereo rig. The parameters of the evolutionary operators of mutation and crossover are as follows: mutation $\eta_m = 25$ and crossover $\eta_x = 2$. Note that the last two parameters describe how the evolutionary operations are applied, while the rates of mutation and crossover specifies how many individuals are generated with those operations.

The advantage of using the honeybee search algorithm is the robustness against outliers. We can appreciate in the *VRML* images of Figure 4 that all 3D points are grouped coherently with the goal of reconstructing compact patches. This is due to the intelligent process described in this paper in which some artificial honeybees (explorers) guide the search process to obtain an improved sparse reconstruction. The explorers guide the foragers using texture and correlation information during the whole process. Similar to the natural process the goal is achieved using a communication system that we have adapted to the classical evolutionary algorithm. It is suitable to think that the honeybee search algorithm could be applied in other contexts.

Acknowledgments

This research was funded by UC MEXUS-CONACYT Collaborative Research Grant 2005; through the project "Intelligent Robots for the Exploration of Dynamic Environments". Second author supported by scholarship 0179377 from CONACyT. This research was also supported by the LAFMI project.

References

1. A. Boumaza, and J. Louchet. "Dynamic Flies: Using Real-time Parisian Evolution in Robotics". *Applications of Evolutionary Computing.* LNCS 2037, pp. 288-297. Evoworkshops 2001.
2. Collet, P., Lutton, E., Raynal, F., Schoenauer, M., 1999. "Individual GP: an alternative viewpoint for the resolution of complex problems", In: Banxhaf, E., Daida, J., Eiben, A.E., Garzon, M.H., Honovar, V., Jakiela, M. Smith, R.E. (Eds.), Genetic and Evolutionary Computation Conf. GECCO99. Morgan Kaufmann, San Francisco, CA.
3. E. Crist. "Can an Insect Speak? The Case of the Honeybee Dance Language". *Social Studies of Science.* SSS and Sage Publications. 34(1), pp. 7-43.
4. K. Deb. "Multi-Objective Optimization using Evolutionary Algorithms". John Wiley & Sons, Ltd. Baffins Lane, Chichester, West Sussex, PO19 1UD, England. 497 pages.
5. M. Dorigo, V. Maniezzo, and A. Colorni. "Ant System: Optimization by a Colony of Cooperating Agents". IEEE Transactions on Systems, Man, and Cybernetics - Part B. Vol. 26, No. 1, pp. 29-41, 1996.
6. M. Dorigo, G. Di Caro, and L. M. Gambardella. "Ant Algorithms for Discrete Optimization". Artificial Life, Vol. 5, No. 2, pp. 137-172. 1999.

7. E. Dunn, G. Olague, and E. Lutton. “Parisian Camera Placement for Vision Metrology”. to appear Pattern Recognition Letters, Special Issue on Evolutionary Computer Vision and Image Understanding. Olague et al. (eds.). Elsevier Science.
8. K. von Frisch. “The Dance Language and Orientation of Bees”. Cambridge, MA: Harvard University Press.
9. D. E. Goldberg, and J. Richardson. “Genetic Algorithms with Sharing for Multimodal Function Optimization”. *In Proceedings of the First International Conference on Genetic Algorithms and Their Applications.* pp. 41-49. 1987.
10. R.M. Haralick, “Statistical and structural approaches to texture”. Proceeding of the IEEE. 7(5) (1979) pp.786-804.
11. D. Kim “Translating the Dances of Honeybees into Resource Location”. *In Proceedings of the 8th International Conference on Parallel Problem Solving from Nature.* LNCS 3242. pp.962-971, 2004.
12. J. Louchet. “Using an Individual Evolution Strategy for Stereovision”. *Genetic Programming and Evolvable Machines.* 2(2), pp. 101-109. June 2001.
13. J. Louchet, M. Guyon, M. J. Lesot, and A. Boumaza. “Dynamic Flies: A New Pattern Recognition Tool Applied to Stereo Sequence processing”. *Pattern Recognition Letters.* 23(1-3), pp. 335-345. January 2002.
14. G. Olague, B. Hernández, and E. Dunn. “Accurate L-corner Measurement using USEF Functions and Evolutionary Algorithms”. 5th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing. Lecture Notes in Computer Science 2611. pp. 410-421. Springer-Verlag. EvoIASP2003.
15. G. Olague, B. Hernández,and E. Dunn. “Hybrid Evolutionary Ridge Regression Approach for High-Accurate Corner Extraction”. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Madison, Wisconsin, USA, June 16-23, 2003. Vol. 1, pp. 744-749.
16. G. Olague, F. Fernndez, C. B. Prez, and E. Lutton. “The Infection Algorithm: An Artificial Epidemic Approach for Dense Stereo Matching”. *In Proceedings of the 8th International Conference on Parallel Problem Solving from Nature.* LNCS 3242. pp.622-632, 2004.
17. G. Olague, and B. Hernández. “A New Accurate and Flexible Model-based Multicorner Detector for Measurement and Recognition”. Pattern Recognition Letters. Volume 26, Issue 1 , 1 January 2005, Pages 27-41.
18. G. Olague, F. Fernndez, C. B. Prez, and E. Lutton. “The Infection Algorithm: An Artificial Epidemic Approach for Dense Stereo Correspondence”. to appear Artificial Life, MIT Press.
19. M.V Srinivasan, S.W. Zhang and H. Zhu. “Honeybees link sights to smells”. Nature (Lond), Vol. 396, pp. 637-638. 1998.
20. M.V. Srinivasan, S.W. Zhang, M. Altwein, and J. Tautz. “Honeybee navigation: nature and calibration of the odometer”. Science, Vol. 287, pp. 851 - 853. 2000.
21. P. K. Visscher. “Dance Language”. *Encyclopedia of Insects.* Academic Press. V. H. Resh and R. T. Carde (eds.), 2003.

Improving the Segmentation Stage of a Pedestrian Tracking Video-Based System by Means of Evolution Strategies*

O. Pérez, M.Á. Patricio, J. García, and J.M. Molina

Universidad Carlos III de Madrid. Computer Department,
Avenida de la Universidad Carlos III, 22 Colmenarejo Madrid. 28270. Spain
{opconcha, mpatrici, jgherrer}@inf.uc3m.es, molina@ia.uc3m.es
http://www.giaa.inf.uc3m.es/

Abstract. Pedestrian tracking video-based systems present particular problems such as the multi fragmentation or low level of compactness of the resultant blobs due to the human shape or movements. This paper shows how to improve the segmentation stage of a video surveillance system by adding morphological post-processing operations so that the subsequent blocks increase their performance. The adjustment of the parameters that regulate the new morphological processes is tuned by means of Evolution Strategies. Finally, the paper proposes a group of metrics to assess the global performance of the surveillance system. After the evaluation over a high number of video sequences, the results show that the shape of the tracks match up more accurately with the parts of interests. Thus, the improvement of segmentation stage facilitates the subsequent stages so that global performance of the surveillance system increases.

1 Introduction

Surveillance systems are usually made up by several interconnected processing blocks or stages that form a high-level representation of the sensed world. The optimization of a video surveillance system consists of improving the particular performance of a stage of the system by adding new computations and adjusting the parameters which run this stage, so that the whole system increases its global performance. In [1], the authors showed how to construct and tune a multi-stage video surveillance system to obtain a good performance in the tracking of aircraft and vehicles moving in an airport surface [2].

In this work, we adapt the system to track people based on the same architecture of the tracking system for surface surveillance in airports. The first new problem that arises from this adaptation is that the parts of interest or blobs appear more fragmented as people are less compact (especially for the extremities) than aircraft or vehicles [3]. Second, one of the main drawbacks of

* Funded by CICYT (TIC2002-04491-C02-02).

outdoor motion estimation is shadows [4] - [6], which attached to the moving people make the system deform the real target. Moreover, shadows might [7] interferer in subsequent stages of the tracking system so that it would be desirable to remove them in a previous phase. The purpose of this study is to improve the performance of this segmentation stage by adding a new block so that the system detects more compact people-shaped blobs and eliminates if possible the shadows so that the total performance of the whole system increases. The parameters that regulate this new block will be searched and tuned by an Evolution Strategy (ES), which has proved to be valid for this type of problems in works like [1]. The main goal of this work is to present based upon the idea that most morphological image analysis tasks, can be reframed as image filtering problems and that ES can be used to discover optimal filters which solve such problems. This paper also introduces the fitness function that assesses the performance of the segmentation block [8, 9]. Finally, we must test the improvement on the complete system for which we need an evaluation function. Although there is a large literature and previous works on metrics for performance evaluation [1], [10]-[12], this paper shows an original proposal based on a minimal *ground truth* record and it is able to evaluate a large number of video samples to obtain significants statistical results.

This paper attempts to address these points. First, section 2 presents a study of the segmentation stage in our video surveillance system. Section 3 details the main problems that we face on tracking people and the solutions adopted. Then, the evaluation function to assess the global performance of the system is presented in Section 4. The details of the experiments and the final conclusions are given in Sections 6.

2 Segmentation Stage

Automated visual surveillance aims to provide an attention-focussing filter to enable an operator to make an optimum decision whenever an unusual event occurs. This is achieved by directing the operators attention only to those events classified as unusual. A generic video processing framework for automated visual surveillance system [13, 14] is depicted in Figure 1. Although some stages require interchange information with others, this framework provides a good structure for the comprehension of our work.

A relevant problem in computer vision is the detection and tracking of moving objects in video sequences. The detection of moving objects can be difficult for several reasons. We need to account for possible motion of the camera, changes in illumination of a scene and shadows, objects such as waving trees, objects that come to a stop and move again such as vehicles at a traffic light, etc. Once the moving objects have been identified, tracking them through the video sequence can also be difficult, especially when the objects being tracked are occluded by buildings or moved in and out of the frame due to the motion of the camera.

By Segmentation Stage, we mean the task of detecting regions that correspond to moving objects such as people and vehicles in video. This is the first basic step

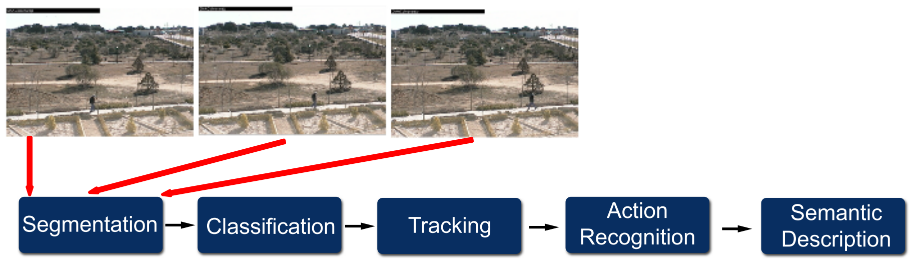


Fig. 1. A generic video processing framework for automated visual surveillance system

of almost every vision system since it provides a focus of attention and simplifies the processing on subsequent analysis steps. As we have said above, due to dynamic changes in natural scenes such as sudden illumination, shadows, weather changes, motion detection is a difficult task to process reliably. Frequently used techniques for moving object detection are background subtraction, statistical methods, temporal differencing and optical flow. Most of the moving object detection techniques are pixel based [16, 17]. Background subtraction techniques attempt to detect moving regions by subtracting the current image pixel-by-pixel from a reference background image that is created by averaging images over time in an initialization period. The pixels whose difference exceeds a threshold are classified as foreground. Although background subtraction techniques perform well at extracting most of the relevant pixels of moving regions, they are usually sensitive to dynamic changes when, for instance, repetitive motions (tree leaves moving in windy day, see Figure 2), or sudden illumination changes occur.

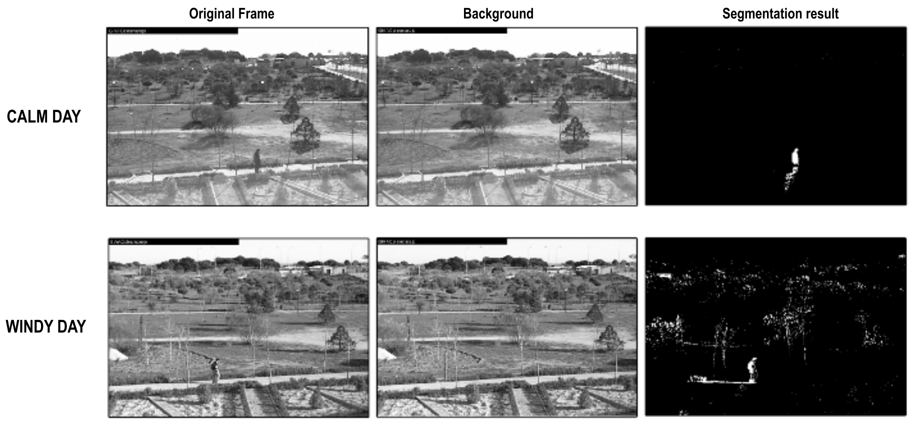


Fig. 2. Different segmentation results obtained in different condition. The first row shows the excellent segmentation results in a calm day. However in the second row, due to the tree leaves in a windy day, we observe brightness changes almost everywhere in the image. Thus, the segmentation stage obtains worse performances.

More advanced methods that make use of the statistical characteristics of individual pixels have been developed to overcome the shortcomings of basic background subtraction methods. These statistical methods are mainly inspired by the background subtraction methods in terms of keeping and dynamically updating statistics of the pixels that belong to the background image process. Foreground pixels are identified by comparing each pixels statistics with that of the background model. This approach is becoming more popular due to its reliability in scenes that contain noise, illumination changes and shadow [14, 18]. Temporal differencing attempts to detect moving regions by making use of the pixel-by-pixel difference of consecutive frames (two or three) in a video sequence [19].

2.1 Background Subtraction

In our system we have implemented a background subtraction approach [20]. The segmentation algorithm is based on the detection of targets contrasting with local background, whose statistics are estimated and updated in an auxiliary image, Background. Then, the pixel level detector is able to extract moving features from this static background, simply comparing the difference with a threshold:

$$Detection(x,y) = [Im(x,y) - Back(x,y)] > THRESHOLD * \sigma \tag{1}$$

where σ represents the standard deviation of pixel intensity. A low threshold would mean a higher sensitivity value, leading to many false detections and higher probability of detection and not corrupting target shape quality. This is one of the key parameters of the system. The background statistics (mean and variance) for each pixel are estimated, from the sequence of previous images, with a simple iterative process and weights to give higher importance to the most recent frames. Besides, in order to prevent targets from corrupting background statistics, the update is just performed for pixels not too near of a tracked target, using the tracking information in the detector. So, the statistics for k-th frame are updated as:

$$\begin{aligned} &Back(x,y,k) = \alpha Im(x,y,k) + (1-\alpha)Back(x,y,k-1) \\ &\sigma^2(x,y,k) = \alpha[Im(x,y,k) - Back(x,y,k-1)]^2 + (1-\alpha)\sigma^2(x,y,k-1) \end{aligned} \tag{2}$$

being x and y pixels out of predicted tracks.

In Figure 3 some segmentation results are depicted following this approach.

2.2 Morphological Post-processing

As we can see in Figure 3, the last step (labelled as 'Segmentation result') obtains excellent results. However, it seems obvious that we can improve the segmentation stage. A pedestrian zoom views of the Figure 3 are depicted in Figure 4. The white pixels make up the pedestrian and set up the foreground pixel map, in which there are unconnected and missing areas. Furthermore, in all over Figure 3 there is a lot of noise which can confuse later processing. The goal of the segmentation stage is not only to produce foreground pixel maps as accurately

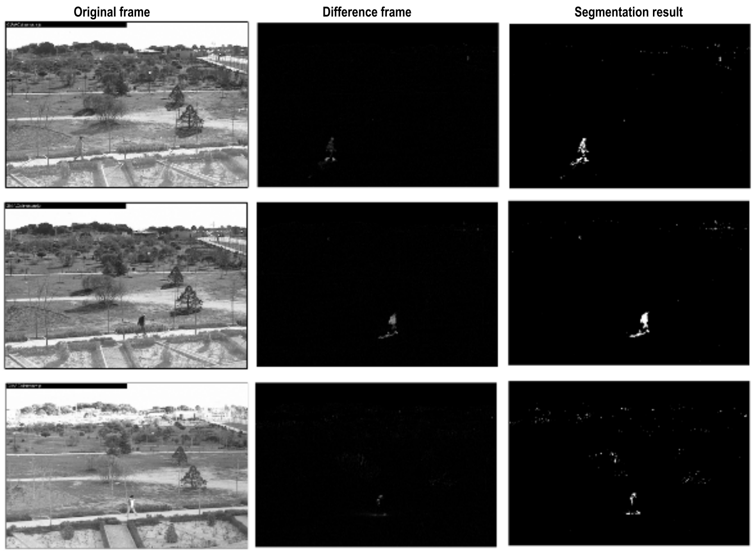


Fig. 3. Instances of the segmentation stage. Although the results are good enough (third column), notice that in the third row, the object detected is rather difficult to track.

as possible, e.g. by removing the special types of noise, but rather to make the pedestrians segmentation more visible and easier to process in the classification stage (see Figure 1).

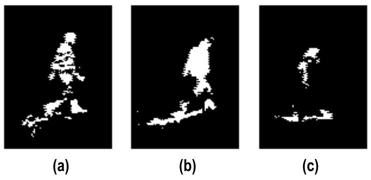


Fig. 4. A pedestrian zoom views of Figure 3. It is clear that we can improve the segmentation stage. In the images appear unconnected and missing areas.

In order to improve segmentation results, morphological operators have been implemented. The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory. Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. The most basic morphological operations are dilation and erosion. In a morphological operation, the value of each pixel in the output image is based on a comparison between the corresponding pixel in the input image and its neighbors. By choosing the size and shape of the neighborhood, a morphological operation can be tuned to be sensitive to specific shapes in the input image. In our case, an erosion operator has been chosen as first post-processing step in order to remove the noise. Then, we apply a delation operator to improve the size and shape of the pedestrian.

Now, our problem is concerned with the selection of the size of the suitable structuring element and the number of iterations of the erode and dilate operations. We define the rectangular size of structuring elements and the number of iteration of erosion and dilate process by the next parameters: HORIZONTAL-SIZE-ERODE, VERTICAL-SIZE-ERODE, HORIZONTAL-SIZE-DILATE, VERTICAL-SIZE-DILATE, ITERATIONS-NUMBER-ERODE and ITERATIONS-NUMBER-DILATE. Besides, we have to establish another parameter involving in the segmentation stage: the THRESHOLD in Equation 1. The election of the values of these parameters makes a big difference in the performance of the system. Thus, in the next section, we show how to use Evolution Strategies in order to optimize these parameters.

3 Optimizing Morphological Parameters by Means of Evolution Strategies

Evolutionary Computation (EC) comprises several robust search mechanisms based on underlying biological metaphor. Having been established as a valid approach to problems requiring efficient and effective search, EC are increasingly finding widespread application in business, scientific and engineering circles. Not much work has been applied in automatic visual surveillance systems using EC. Perhaps the main trouble is related with the enormous amount of data to process. In [21] genetic programming is used to segment video sequences. Hwang [22] shows a genetic algorithm which uses both spatial and temporal information to segment and track moving objects in video sequences. In [1], an Evolution Strategy (ES) for optimizing the parameters regulating a video-based tracking system is presented.

We have implemented ES for improving the segmentation stage by adjusting the parameters listed above. Regarding the operators, the type of crossover used in this work is the discrete one and the replacement scheme which is used to select the individuals for the next generation is $(\mu + \lambda) - ES$.

In an ES, the fitness is a function that gives a *score* to the outcome of the system and its design is probably the most critical task concerning both the domain problem and the ES itself. In fact, it must be based on the foreground pixel map's features and most of the parameters within this domain algorithm could affect the outcome of the segmentation stage.

After the morphological post-processing of an image, its foreground pixel map consist of several blobs (i.e. coherent connected regions). In order to simplify the process, we represents the blobs by its bounding rectangle. Let NB be the Number of Blobs in a foreground pixel map. In our experimentation we have been working with videos where there is only a pedestrian, and therefore we expect to found a short number of blobs in our ideal segmentation stage.

Let Im and $\widehat{Im}$ be the image before and after the morphological post-processing, respectively. We define $Im(x, y)$ and $\widehat{Im}(x, y)$ as `true`, if and only if the pixel (x, y) belongs to a moving object, respectively. We define the Density ratio, $D(B)$, of a blob, B, as:

$$D(B) = \frac{1}{n} \quad Card\{Im(x, y) \quad \wedge \quad \widehat{Im}(x, y)\}; \qquad \forall(x, y) \in B. \tag{3}$$

where n is the number of pixels in the blob B and *card* stands for the cardinality (i.e. number of pixels) of a set. The operator $'\wedge'$ (and) is applied to assess which part of the processed image contains detected pixels in the original image.

Let $AR(B)$ the Aspect Ratio of a blob, B. A blob is represented by its bounding rectangle. We define $AR(B)$ as:

$$AR(B) = \frac{width(B)}{height(B)} \tag{4}$$

where *width* and *height* stands for the bounding rectangle's width and height of a blob, respectively. Since, in our system, pedestrians are the object that we have to track, in contrast of shadows or noise, we expect to get a small value for the AR(B) ratio in every blob.

At last, the fitness function that we have to minimize is:

$$fitness = \alpha NB + \beta \sum_{\forall B \in \widehat{Im}} AR(B) - \gamma \sum_{\forall B \in \widehat{Im}} D(B) \tag{5}$$

where α, β and γ are normalization coefficients.

4 Evaluation System

The main requirement for surveillance systems is the capability for tracking objects of interests in operational conditions, with satisfactory levels of accuracy and robustness. The difficult task is the definition of an automatic, objective and detailed procedure able to capture the global quality of a given system in order to support design decisions based on performance assessment. There are many studies that evaluate video surveillance systems against the ground truth or with synthetic images. Our contribution is a new methodology to compute detailed evaluations based on a minimal amount of hand-made reference data and a large quantity of samples. The result is a robust assessment based on a statistical analysis of a significant number of video sequences. Thus, our work used the proposed evaluation system to assess the surveillance system and check the increase of the total performance.

4.1 Basis of the Evaluation System

The system requires as reference a function *f(x,y)* (it could be a function defined on parts) that describes as well as possible the mean track followed by the targets we want to track. In this case, we have recorded a set of video sequences of people walking along a footpath. Our set of samples was divided into two groups: (1) 50 video sequences of people moving from right to left along a footpath, and (2) 50 video sequences of people moving from left to right along a footpath.

Thus, the subsequent assessment was separated into two steps and we obtained two sets of results for each one of the video sequences.

The function *f(x,y)* that approximates the objects' trajectory was very simple in both cases. It was a straight line that was considered the ground truth for the system.

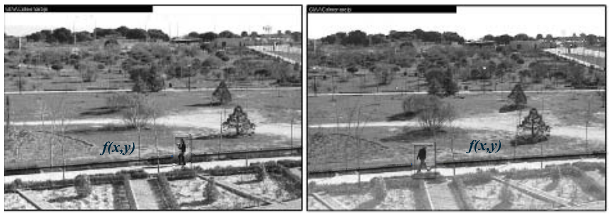


Fig. 5. Video shot samples from the two sets of sequences and the function f(x,y) that approximates the trajectory of each pedestrian

4.2 Evaluation Metrics

This section explains the core of the evaluation system and how it worked in our experiments. The evaluation system collected the tracks that were given by the tracking system for each frame in all the video sequences. Then, a distance to the reference function *f(x,y)* was computed so that only the tracks whose distance was less than a given margin were considered for the subsequent assessment. The set of metrics considered for this particular problem are listed below:

Absolute Area. It is computed by calculating the area of the detected track.

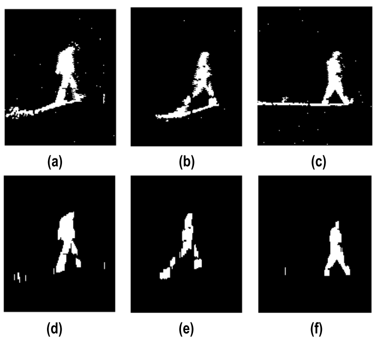


Fig. 6. Segmentation results before (from (a) to (c)) and after (from (d) to (f)) the morphological post-processing

Transversal Error. It is defined as the distance between the center of the track and the segment which is considered as ground truth in this moment.

Continuity Faults. This metric checks if a current track existed the previous moment or did not. If the track did not exist, it means that this track was lost by the tracking system and recovered in a subsequent frame. This behavior must be computed as continuity fault. This continuity metric is a counter where one unit is added each time a continuity fault occurs.

Changes of Direction. This metric marks when a track changes its direction. This metric is also a counter where one unit is added each time a change of direction occurs.

5 Methodology and Results

Our general procedure for processing the video sequences was as follows:

1. Take a set of 5 random videos from the two video sequences groups.
2. Use the evolution strategies for adjusting the parameters of the morphological operators added to the segmentation stage. We implemented ES with a size of 10+10 individuals. This population is the minimum that assures the same result as if we had taken a higher number of individuals. The mutation factor of $\triangle\sigma = 0.5$ and the initial seed was fixed at 100.
3. Repeat the experiment with at least three different seeds.
4. If the results are similar, fix the parameters of the morphological algorithms for using them in all videos.
5. Take one video sequences set and the parameters obtained by the evolution strategy. Make the surveillance system work and collect all the people's tracks for each frame of each video sequence.
6. Evaluate these tracks and compare the results (with and without morphological algorithms in the segmentation stage).
7. Repeat the process for the second set of videos sequences from step 5.

In order to compare the effect of the morphological operators, we show some pictures before and after the application of the algorithms (Figure 6). The results of the optimization parameters are shown in Table 1. We can observe that the

Table 1. Optimization results. Notice that the structuring element shape rewards high and thin objects according to the pedestrians' shape.

HORIZONTAL-SIZE-ERODE	1
VERTICAL-SIZE-ERODE	4
HORIZONTAL-SIZE-DILATE	1
VERTICAL-SIZE-DILATE	4
ITERATIONS-NUMBER-ERODE	2
ITERATIONS-NUMBER-DILATE	2
THRESHOLD	15

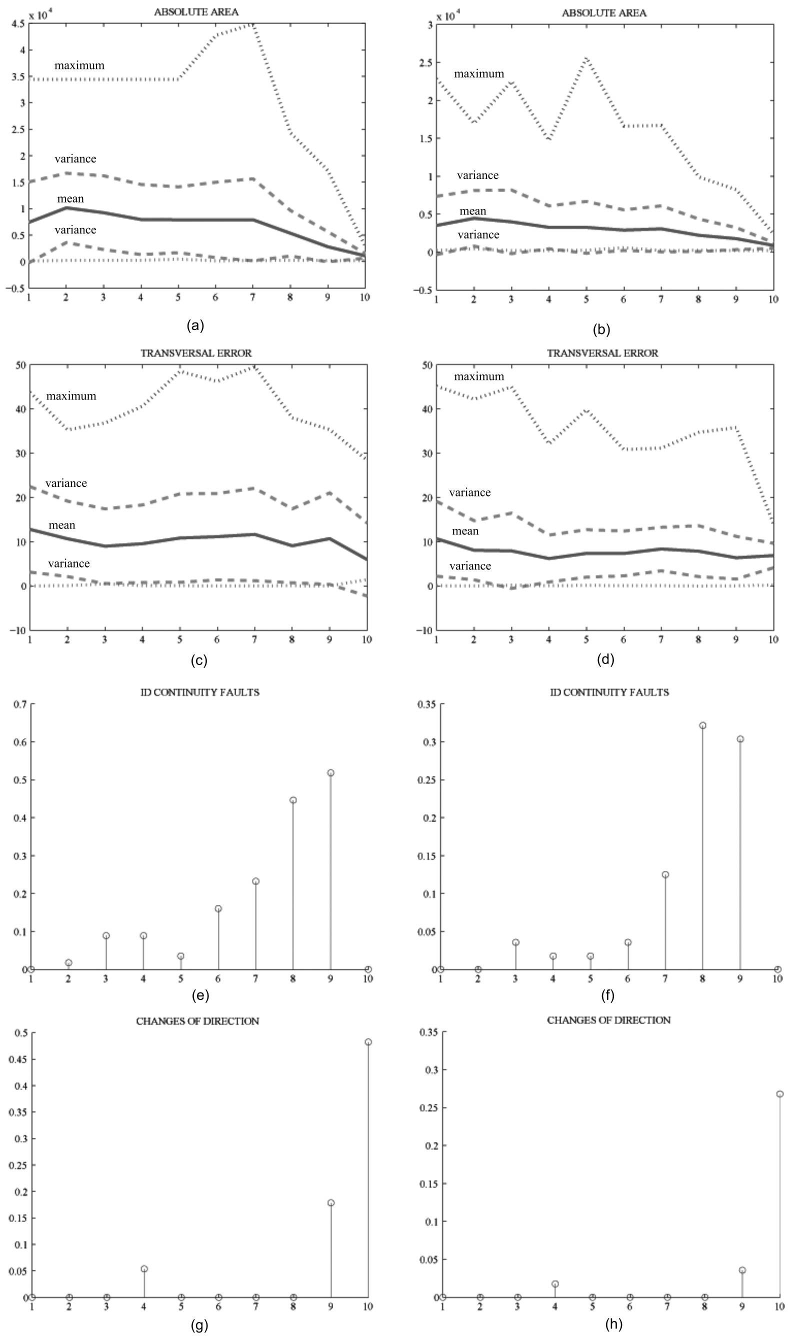


Fig. 7. Metrics for people walking from right to left before and after the morphological process (left and right column respectively)

Table 2. Numerical statistics of the Absolute Area and Transversal Error

	Before morphological operators			After morphological operators		
	Mean	Max	Min	Mean	Max	Min
Absolute Area	7033	44890	111	3057.7	25636	203
Transversal Err.	10.5	49.5	0.009	7.8	45.15	0.00055

shadows and noises were removed so that subsequent stages of the surveillance system created more appropriate tracks according to the parts of interests. That is, the results had a real correspondence between the people we were interested in and the resulted tracks of the tracking system. This affected directly the track size, which was smaller as a consequence of the shadow elimination. This effect is displayed in the Table 2.

Finally, the effect on the whole surveillance system is showed in Figure 7. In order to have a more detailed idea of the system performance, the area under study is divided into 10 zones. Each zone is defined as a fixed number of pixels of the x-axis, the 10% of the horizontal size of the image. The absolute area and the transversal error show the mean, variance and maximum values for each of these two metrics.

All the metrics presented a remarkable improvement on the behavior of the total surveillance system. The absolute area decreased its mean value from 7033 to 3057.7 (see Table 2 and Figure 7(a) and 7(b)) due to the better adjustment of the tracks to the pedestrian shape. Second, the transversal error improved from a mean value of 10.5 to 7.8, which means that the gravity center of the people's track is closer to the ground truth function $f(x, y)$. Moreover, the last figures show that the number of losses for the tracks and the changes of direction decreased by a factor of 2.

As a final conclusion, we are able to confirm that the improvement in the segmentation stage provides more compact and accurate blobs to the subsequent blocks of the video surveillance system so that the performance of the surveillance system does increased.

References

1. Pérez, O., García, J., Berlanga, A., and Molina, J.M.: Evolving Parameters of Surveillance Video System for Non-Overfitted Learning. Proc. 7^{th} European Workshop on Evolutionary Computation in Image Analysis and Signal Processing. EvoIASP 2005. Lausanne, Switzerland (2005).
2. García, J. A. Besada, J. M. Molina, J. Portillo. Fuzzy data association for image-based tracking in dense scenarios, IEEE International Conference on Fuzzy Systems. Honolulu, Hawaii (2002)
3. Friedman, N. and Russell, S.: Image segmentation in video sequences: A probabilistic approach, in Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 97), Morgan Kaufmann Publishers, Inc., (San Francisco, CA) (1997) 175–181.
4. Rosin, P.L., and Ellis, T.: Image Difference Threshold Strategies and Shadow Detection, in the 6th BMVC 1995 conf. proc., Birmingham, UK, (1995) 347–356.

5. Jiang, C.: Shadow identification, CVGIP: Image Understanding, **59(2)** (1994) 213–225.
6. Prati, A., Mikic, I., Trivedi, M.M., Cucchiara, R.: Detecting Moving Shadows: Algorithms and Evaluation, IEEE Trans. PAMI **25(7)** (2003) 918–923.
7. Bevilacqua, A.: Effective Shadow Detection in Traffic Monitoring Applications. WSCG 2003, **11(1)**
8. Zhang, Y.J.: Evaluation and comparison of different segmentation algorithms, Pattern Recognition Letters **18** (1997) 963–974.
9. Chabrier, S., Emile, B., Laurent, H., Rosenberger, C.,March, P.: Unsupervised Evaluation of Image Segmentation Application to Multi-spectral Images. 17th International Conference on Pattern Recognition (ICPR'04) **1** (2004) 576–579.
10. Erdem, E., Sankur, B., Tekalp, A.M.: Metrics for performance evaluation of video object segmentation and tracking without ground-truth. ICIP **2** (2001) 69–72.
11. Pokrajac, D. and Latecki, L.J.: Spatiotemporal Blocks-Based Moving Objects Identification and Tracking. IEEE Int. W. Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS). Nice, France. (2003).
12. Black, J., Ellis, T., and Rosin, P.: A Novel Method for Video Tracking Performance Evaluation, Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS). Nice, France. (2003).
13. Aggarwal, J.K. and Cai, Q.: Human motion analysis: A review. Computer Vision and Image Understanding, **73(3)** (1999) 428-440.
14. Wang, L., Hu, W., and Tan., T.: Recent developments in human motion analysis. Pattern Recognition, **36(3)** (2003) 585-601.
15. Haritaoglu, D.I., Harwood, D., and Davis, L.: W4: Real- Time Surveillance of People and Their Activities, IEEE Trans. Pattern Analysis and Machine Intelligence **22(8)** (2000) 809-830.
16. Remagnino, P., Jones, G.A., Paragios, N., and Regazzoni, C.S.: Video-Based Surveillance Systems. Kluwer Academic Publishers, 2002.
17. Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A.P.: Pfinder: Real-time Tracking of the Human Body, IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) **19(7)** (1997) 780-785.
18. Stauffer, C., and Grimson, W.: Adaptive background mixture models for realtime tracking. In Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (1999) 246–252.
19. Lipton, A.J., Fujiyoshi, H., and Patil, R.S.: Moving target classification and tracking from real-time video. In Proc. of Workshop Applications of Computer Vision, (1998) 129-136.
20. Cohen, I., and Medioni, G.: Detecting and Tracking Moving Objects in Video from an Airborne Observer, Proc. IEEE Image Understanding Workshop, (1998) 217–222.
21. Kim, E.Y., Park, S.H., Hwang, S., and Kim, H.J.: Video Sequence Segmentation Using Genetic Algorithms. Pattern Recognition Letter, **23(7)** (2002) 843–863.
22. Hwang, S., Kim, E.Y., Park, S.H., and Kim, H.J.: Object Extraction and Tracking Using Genetic Algorithms, in Proc. IEEE Signal Processing Society ICIP **2** (2001) 383–386.

An Adaptive Stochastic Collision Detection Between Deformable Objects Using Particle Swarm Optimization

Wang Tianzhu, Li Wenhui, Wang Yi, Ge Zihou, and Han Dongfeng

Key Laboratory of Symbol,
Computation and Knowledge Engineering of the Ministry of Education,
College of Computer Science and Technology,
Jilin University, Changchun, 130012, P.R. China
wangtz@56.com, liwh@public.cc.jl.cn

Abstract. In this paper, we present an efficient method for detecting collisions between highly deformable objects, which is a combination of newly developed stochastic method and Particle Swarm Optimization (PSO) algorithm. Firstly, our algorithm samples primitive pairs within the models to construct a discrete binary search space for PSO, and in this way user can balance performance and detection quality. Besides a particle update process is added in every time step to handle the dynamic environments caused by deformations. Our algorithm is also very general that makes no assumptions about the input models and doesn't need to store additional data structures either. In the end, we give the precision and efficiency evaluation about the algorithm and find it might be a reasonable choice for complex deformable models in collision detection systems.

1 Introduction

Fast and accurate collision detection between deformable geometric bodies is essential in application areas like virtual reality, animation, simulation, games and robotics. Numerous approaches have been investigated to it [1]. Recently, due to observations that perceived quality of most interactive 3D applications does not depend on exact simulation, but rather on real-time response to collisions [1] and that humans cannot distinguish between physically-correct and physically-plausible behavior of objects [2], stochastic collision detection becomes a focus, which can trade off accuracy for computation time by selecting random primitives (vertices, edges, triangles, etc.) as an initial guess of the potential collided regions within two meshes. In this way, collision detection is converted to the problem to search through sample space to find primitive pairs whose distance are shorter than the given threshold as fast as possible.

This paper proposes the use of the Particle Swarm Optimization (PSO) algorithm to solve the problem described above. PSO is a population based evolutionary algorithm developed by Kennedy and Eberhart [3] [4] [5] and has been compared to genetic algorithms [6] for efficiently finding optimal or near-optimal solutions in large search spaces. Besides, PSO is an adaptive algorithm that leads itself very well to dynamic changes and has great capability in solving the problem described above for stochastic collision detection.

Most of collision detection algorithms are heavily based on data structures that can be more or less pre-computed and update during every simulation steps. In our algorithm, we make no assumption about the input model, which can without topology or with changing topology and even be “polygon soups”. Besides, the algorithm needn’t to store additional data structures such as bounding volume hierarchies, so the memory cost is low. It doesn't make any assumptions about object motion or temporal coherence between successive frames either. The model primitives may undergo any motions, vertices or edges can be inserted or deleted.

The remainder of the paper is organized as follows. We survey some related work on collision detection for deformable objects in Sec.2. Sec. 3 shows the unified PSO algorithm. Sec. 4 gives an overview of our approach and we give more detail of our algorithm to handles deformable models in Sec.5. In Sec.6, we give precision and efficiency evaluation about the algorithm and discuss some of its limitations.

2 Related Work

Numerous collision detection methods have been extensively studied, such as Spatial partitioning method, Bounding Volume Hierarchies, Image-Space techniques, GPU-based techniques, Distance Fields, or combination of them. Usually these mentioned methods have been demonstrated to work efficiently in different kinds of environments for rigid body simulations. But when we consider the problem of deforming bodies, they are not that useful, as they rely heavily on pre-computed data and data structures or they are dependent on certain body characteristics, for example, bodies that must be decomposed into convex pieces. A very general collision detection method for deformable objects has been proposed by Smith [7]. At every time step, the AABB of all objects is calculated. When two overlapping AABBs are found, object faces are first pruned against their overlap region. Remaining faces from all such overlap regions are used to build a world face octree, which is traversed to find faces located in the same voxels. A data structure called the BucketTree has also been proposed [8], which is an octree data structure with buckets as leaves where geometrical primitives can be placed. Another approach is suggested by van den Bergen [9], which is also used in the collision detection library called SOLID[10]. Initially, AABB trees are built for every model in its own local coordinate system. The AABBs in the trees are then transformed as the models are moved or rotated in the scene. This transformation causes the models’ locally defined AABBs to become OBBs in world space. When a model is deformed an update of the affected nodes in the trees has to be done. In the literature, there are also some other algorithms for flexible objects. For a more detailed review, please refer to the survey paper by [1].

Recently, “inexact” methods have become a focus in collision detection research. Stochastic methods selects random pairs of colliding features as an initial guess of the potential intersecting regions, and by this way it trades-off accuracy for computation time. When the object moves or deforms, the method considers temporal coherence in the sense that if a pair of features is close enough at a time step, it may still be interesting in the next one [1]. [11]uses probabilistic principles to estimate the possibility of collision with respect to a given quality criterion. This approach is restricted to rigid objects. [12]combines BVHs and stochastic sampling: They use hierarchy of bounding volumes (k-dops) to pick appropriate random geometric primitives pairs and then chosen pairs iteratively converge to local distance minima. They also use spatial and space coherence to keep track of this local minimum over the neighborhood features. But it needs topology information and hierarchy structures.

Our work inspires from ideas of above and aims to accelerate the searching process by Particle Swarm Optimization. The algorithm design, however, is problem of specifying and ascertaining many implicit and explicit factors, which have great effects to the performance. In the next sections, the design of the PSO for stochastic collision detection is described, and the results demonstrate this method leads to a more robust collision detection system.

3 Unified Particle Swarm Optimization

PSO is a new optimization technique originating from artificial life and evolutionary computation. Different from most commonly used population-based evolutionary algorithms, such as genetic algorithms, evolutionary programming, evolutionary strategies and genetic programming, PSO algorithm is motivated from the simulation of social behavior. The technique involves simulating social behavior among individuals (particles) "flying" through a multidimensional search space and evaluating particles' positions relative to a goal (fitness) at every iteration. Particles in a local neighborhood share memories of their "best" positions, and use these memories to adjust their own velocities, and thus subsequent positions [4].

The original formulae developed by Kennedy and Eberhart[3] was improved by Shi and Eberhart [4]. Particle i is represented as $Xi=(x_{i1},x_{i2},...,x_{iD})$ in D-dimensions. Each particle maintains a memory of its previous best position $P_i=(p_{i1},p_{i2},...,p_{iD})$ and a velocity along each dimension $V_i=(v_{i1},v_{i2},...,v_{iD})$. In each iteration step, the P vector of the particle with the best fitness in the local neighborhood, designated g, and the P vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle. An inertia parameter ω, that is used to increases the overall performance of PSO [5]. These formulae are:

$$v_{id} = \omega * v_{id} + \eta_1 * \text{rand}_1() * (p_{id} - x_{id}) + \eta_2 * \text{rand}_2() * (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where constants η_1 and η_2 determine the relative influence of the social and cognitive components, and are usually both set the same and give each component equal weight as the cognitive and social learning rate [4].$\text{Rand}_1()$ and $\text{rand}_2()$are two random functions in the range [0,1].

PSO algorithm has the ability to complete finding optimal or near-optimal solutions in large search spaces efficiently, which is very suitable for stochastic collision detection. Furthermore, PSO algorithm also has the advantages that it is easy to overcome local optimization problem and process constrained conditions. PSO also can be implemented with ease and few parameters need to be tuned. In the next section, we will introduce our notation and give an overview of our approach.

4 Algorithm Overview

In the beginning we convert 3-Dimensions object space to discrete binary search space ($\Omega^3 \rightarrow \Phi^2$) for adopting PSO.

Two positive integer sets A and B are set, containing the serial number of primitives (here only point-point pairs are considered), which respectively belong to two models meshes. And a particle represents a pair of primitives between two sets. So in search space, the position of each particle is represented by AB-axis intersection and can be notated as Equation (3). The velocity is expressed by v_a (the velocity of A-axis) and v_b (the velocity of B-axis). Moreover, primitive pairs are antithetical, the search space can be half.

$$p_k=\{A_i,B_j\} \tag{3}$$

Each particle also has a fitness, which is the distance between them in the 3-dimensions object space described by Equation (4):

$$f=\sqrt{(a.x-b.x)^2+(a.y-b.y)^2+(a.z-b.z)^2} \tag{4}$$

where a and b represent two vertexes, and each vertex has a (x, y, z) coordinate in 3-dimensional object space.

In order to reduce computation, we use the square of particle's fitness. From above design, collision detection is converted to search through a discrete binary space to find those particles (optimal solution) whose fitness is below a given proximity threshold.

For the case, the optimal solution is single the swarm will probably converge to it after iterations. But if the optimal solution is a set, such as collisions between sawtoothed models, or a collision cluster, how can we find them as many as possible? Here, we'd better look back to the settings of search space. At beginning we set it without taking primitives' topology constraints into account, so continuous primitives in object space may distribute discretely in search space. But that makes our method more robust for various types of models. Here, we make use of one outside collided pairs set to record particles those falling below a given proximity threshold in each iteration. With the iteration number increasing, more places will be searched and more optima will be found.

Iterations will stop, when maximum iteration number N_I is attained. The maximum iteration within a time step is determined by the Formula(5), which allows users to balance precision (i.e. completeness) and reaction speed (i.e. small time step).

$$N_I=\frac{T}{N_p * C_p} \tag{5}$$

T: A time step; N_P: the population size of the swarm; C_P: one parcel fitness computation cost.

5 Handing Deformable Models

In this section, we give more details of our algorithm in handing collision detection between deformable models.

During the process of simulation, models might deform in every time steps. Deformations of models in most cases are of two types: arbitrary vertex repositioning or splitting. For the former, a model might undergo a complete change of shape, from one time step to the other, by moving the relative position of all of its vertices. But during such deformations, the mesh connectivity stays the same, i.e. the mesh is not torn up in any way. For the latter, a model's geometric primitives might change, such as increasing or decreasing the number or splitting the body into new separated pieces [13].Our method efficiently handles these two kinds of deformation.

To our algorithm, the deformations of model mean variation of search space between time steps. This variation can be both or either of the following: position change of goal or size change of search space.

In the view of the whole simulation, collision detection may be regarded as a consecutive PSO iteration process from the first time step to the last one. Different From the normal PSO, however, search space may undergo a change due to the model deformation after a fixed iteration (a fixed iteration number was deigned in a time step). This is a dynamic environments problem.

PSO has been proven to be very effective for dynamic problems. In previous work, resetting particles' memory is used to response to the changes [14]. Carlisle and Dozier investigated using PSO to track a single peak varying spatially [5]. Hu and Eberhart [15] suggested monitoring environments for a change and updating the gbest and pbest values of particles when a change is detected.

As we all know, PSO algorithm has a mechanism to adapt itself to environment changes. The swarm searches for optima in solution space and typically shrink to a small space and if the changes occur in this area, PSO will probably find new optimum automatically without any modification [15]. If the changes are wide, we must take some strategies to response the changes effectively. So in the beginning of each time step (changes might have took place), a particle update process is added, in which fitness of all the particles are valued and a portion of the best of them are reserved. Replace their pbest values by their current X vector. Thus their profits from previous experiences are retained by their current location and they are forced to redefine their relationship to the goal. The other particles are allocated to new positions randomly, that ensure more different place in the search space will be searched, for the case that the whole population has already converged to a small area, it might not easy to jump out to follow the changes. In the section 6, we will test different portion reserved and compare the results.

6 Implementation and Performance

We have done many different experiments to investigate the performance characteristics of our proposed method. The experiments used have to be chosen with care, since the results depend on the shapes of the models, their relative orientation and the deforma tions applied. All the tests were done using Pentium IV, 2.4 MHz CPU and Memory, 1G.

For the initial experiment, we elected to explore the effects of taking different samples in a static scene of two collided hands (shown in Fig.1). We sampled five groups of primitives, which constituted five search spaces for PSO. It is evident that with more sample pairs, the precision is better, but the search cost is more. For example, if we want a 20% detection ratio, it is better to take group 2or 3 then group 4 or 5. If we want a 30% detection ratio the group 3 is the best. If we want a 60% detection ratio the group 4 is the best. If we want more then 90% detection ratio, group 5 is the only choice (see Table1). So with the different precision demands, we can take different sample strategies to get the best result.

We varied the population from 5 to 100 in a time step compared the success rates (shown in Fig.2).We would expect that, in general, more particles would search more space, and solutions would then be found sooner ,but we found this to be generally true in terms of performance, but not in terms of cost. Because of the time critical, population size and the iteration number are restricted to each other, and when the population increases, each iteration step represents a greater cost, as more particles call upon the evaluation function. A population size from 20 to 30 appears to be good choice. It is small enough to be efficient, yet large enough to produce reliable results.

In the second experiment, two continuously deforming sneaks were moved slowly towards each other during 50 simulation time steps (shown in Fig.3). We test our

Table 1. Compares of detection ratio of five sample groups for of two collided hands

Sample pairs		Detection ratio and time of the sample space			≥ 90% Detection ratio of all
		30%	60%	90%	
1	1000*1000	3 ms	8 ms	10 ms	5.6%
2	5000*5000	6 ms	11 ms	16 ms	19.7%
3	10000*10000	11 ms	17 ms	23 ms	33.4%
4	15000*15000	17 ms	34 ms	59 ms	62.0%
5	26373*26373	33 ms	74 ms	112 ms	92.2%

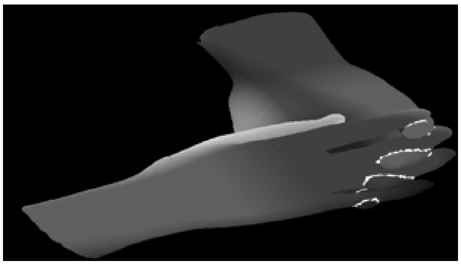


Fig. 1. Scene 1: Two collided hand models each of 26373 primitives and collided pairs are white

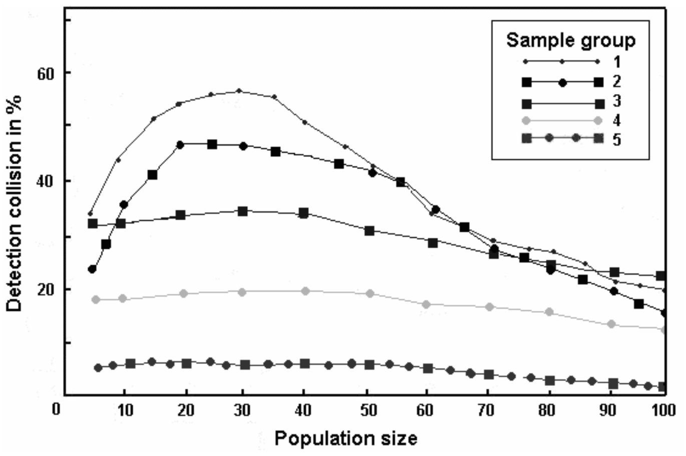


Fig. 2. Detection ratio with varied population size for two collided hands

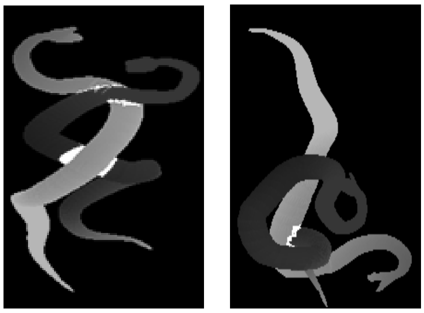


Fig. 3. Scene 2: Simulation of two deforming sneaks (each of 22236 primitives) during 50 time steps and the collided pairs are white

algorithm to the spatio-temporal coherence between consecutive time steps by four updated methods, with a population of 30 particles: (1) Do nothing; (2) Update pbest of all population; (3) Reserve 50% of all and update their pbest, then re-randomize others; (4) Re-randomize all the population. The results are given in Fig. 4 and Fig.5. It's evident that if the deformation is not drastically, to method (1), (2), (3), previous experiences are retained by the particles, and spatio-temporal coherence between consecutive time steps ensure within little iteration, results would be found and detection ratio of method (3) increases faster than that of others. But if the models undergo a drastically deformation, such as a sudden collision took place shown in Fig.5, the method that re-randomize all the population will hold a steadily on ratio, but due to the spatio-temporal coherence, with the time flying, detection ratio of the method(3) became the highest.

At last, we have compared the performance and efficiency of our algorithm against prior collision detection algorithms: AABB and sphere hierarchies for deformable models in scene 2. As the sneaks deforms, many triangles become long and skinny and the bounding volume and sphere become rather large and update process costs a lot of time. Our algorithm obtains speedup due to little number of elementary tests and little update cost. Therefore, as the result shown in the Fig.6, we achieve up to 2-3 times overall speedup. Our algorithm has a limitation, that it couldn't find all the collided pairs in most cases. But Kimmerle showed us in [12] that a detection ratio below 100% is sufficient to ensure a stable simulation. Even for a very challenging cloth simulation, a response ratio as low as 20% can be enough to get good visual results. So to our algorithm, if precision of collision detection is not so high, it leads to a more robust collision detection system.

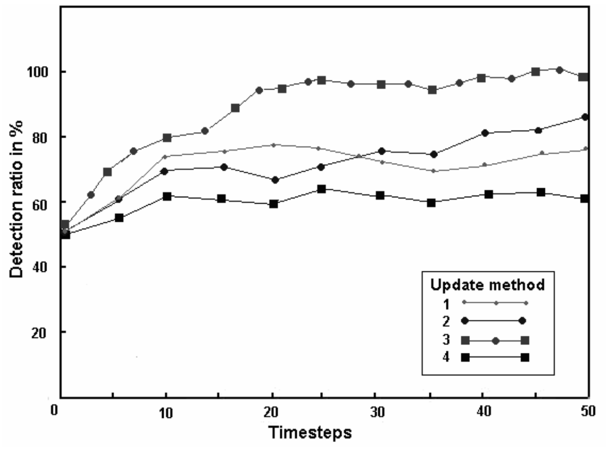


Fig. 4. Comparison of three kinds of update swarm strategies when deformation is not drastically

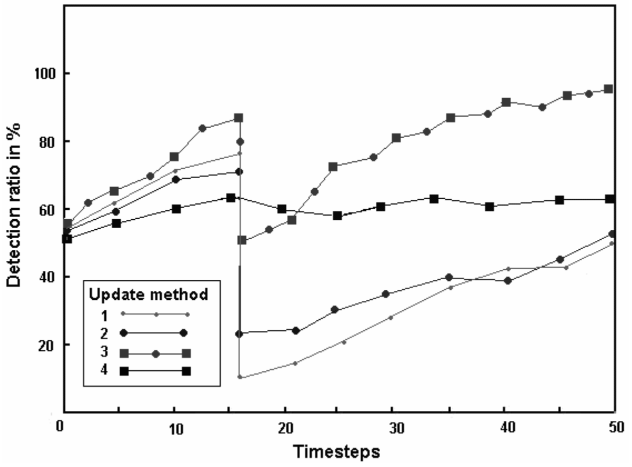


Fig. 5. Comparison of three kinds of update swarm strategies when deformation is drastically

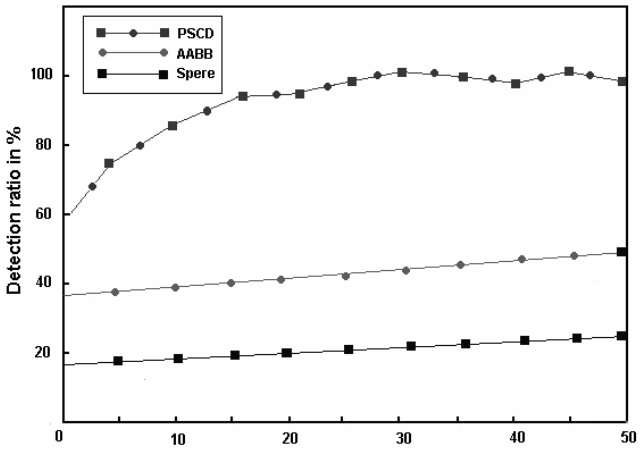


Fig. 6. Comparison of performance of our algorithm (PSCD) with prior-approaches based on AABB and sphere hierarchies

7 Conclusion

This paper has proposed a new method of Stochastic Collision Detection for deformable objects by using PSO Algorithm. Our algorithm samples primitive pairs within the models to construct a discrete binary search space for PSO, by which user can balance performance and detection quality. In order to handle the deformation of models in the object space, a particle update process is added in the beginning of every time step , which handles the dynamic environments problem in search space caused by deformation. This approach provides a more comprehensive way to trade-off accuracy for computation time. Moreover, the swarm can also handle temporal coherence and efficiently search through the highly large primitive pair search space. At last, we give the precision and efficiency evaluation about the algorithm and find it might be a reasonable choice for deformable models in Stochastic Collision Detection.

Acknowledgments

This work was supported by NSFC (No. 60573182).We would like to thank Yan Zhang for discussion on the validation proof and the paper format and other members of the Image& Graphics group at JLU provide much useful feedback on our algorithm.

References

1. Teschner, M., Kimmerle, S., Heidelberger,B.. Collision detection for deformable objects. Eurographics, State of the Art Report(2005).
2. Uno, S., Slater, M.. The sensitivity of presence to collision response. In Proc. of IEEE Virtual Reality Annual International Symposium (VRAIS) (1997).

3. Kennedy, J. and Eberhart, R.. Particle Swarm Optimization, IEEE International Conference on Neural Networks, Perth, Australia(1995).
4. Shi, Y. H. and Eberhart, R. C.. Parameter Selection in Particle Swarm Optimization, The 7th Annual Conference on Evolutionary Programming, San Diego, USA(1998).
5. Carlisle, A. and Dozier, G.. Adapting Particle Swarm Optimization to Dynamic Environments, Proceedings of International Conference on Artificial Intelligence, Las Vegas, Nevada, USA(2000)429–434.
6. Eberhart, R. and Shi, Y.. Comparison between Genetic Algorithms and Particle Swarm Optimization, The 7th Annual Conference on Evolutionary Programming, San Diego, USA(1998).
7. Smith, A., Kitamura, Y., Takemura, H. and Kishino F.. A Simple and Efficient Method for Accurate Collision Detection Among Deformable Polyhedral Objects in Arbitrary Motion. Proceedings of the IEEE Virtual Reality Annual International Symposium(1995)136–145.
8. Ganovelli, F., Dingliana, J. and O'Sullivan.C.. BucketTree: Improving Collision Detection between Deformable Objects. Spring Conference in Computer Graphic, Bratislava (2000)156–163.
9. Van den Bergen, G.. Efficient Collision Detection of Complex Deformable Models using AABB Trees. Journal of Graphics Tools, **2**(4)(1997):1–14.
10. Van det Bergen, G.. SOLID. Software Library for Interference Detection, Available at http://www.win.tue.nl/cs/tt/gino/solid 2 (1999).
11. Klein, J. and Zachmann, G.. Adb-trees: Controlling the error of time-critical collision detection. In Proceedings of Vision, Modeling and Visualization (2003).
12. Kimmerle, S., Nesme, M. and Faure, F.. Hierarchy accelerated stochastic collision detection. In Proceedings of Vision, Modeling, Visualization,(2004) 307–314.
13. Larsson,T., and Akenine-m¨oller.T, Collision detection for continuously deforming bodies. In Eurographics(2001).
14. Eberhart, R. and Shi,Y.. Tracking and optimizing dynamic systems with particle swarm. Proc. Congress on Evolutionary Computation,Piscataway. NJ: IEEE Press (2001)94–97.
15. Hu, X. and Eberhart, R. C..Adaptive particle swarm optimization: detection and response to dynamic systems. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC. IEEE Press (2002) 1666–-1670.

Genetic Programming for Automatic Stress Detection in Spoken English

Huayang Xie, Mengjie Zhang, and Peter Andreae

School of Mathematics, Statistics and Computer Science,
Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
{Huayang.Xie, Mengjie.Zhange, Peter.Andreae}@vuw.ac.nz

Abstract. This paper describes an approach to the use of genetic programming (GP) for the automatic detection of rhythmic stress in spoken New Zealand English. A linear-structured GP system uses speaker independent prosodic features and vowel quality features as terminals to classify each vowel segment as stressed or unstressed. Error rate is used as the fitness function. In addition to the standard four arithmetic operators, this approach also uses several other arithmetic, trigonometric, and conditional functions in the function set. The approach is evaluated on 60 female adult utterances with 703 vowels and a maximum accuracy of 92.61% is achieved. The approach is compared with decision trees (DT) and support vector machines (SVM). The results suggest that, on our data set, GP outperforms DT and SVM for stress detection, and GP has stronger automatic feature selection capability than DT and SVM.

1 Introduction

Stress is a form of prominence in spoken language. Usually, stress is seen as a property of a syllable or of the vowel nucleus of the syllable. There are two types of stress in English. *Lexical stress* refers to the relative prominences of syllables in individual words. *Rhythmic stress* refers to the relative prominences of syllables in longer stretches of speech than an isolated word. When words are used in utterances, their lexical stress may be altered to reflect the rhythmic (as well as semantic) structure of the utterance.

As English becomes more and more important as a communication tool for people from all countries, there is an ever increasing demand for good quality teaching of English as a Second Language (ESL). Learning English well requires lots of practice and a great deal of individualised feedback to identify and correct errors. Providing this individualised feedback from ESL teachers is very expensive, therefore computer software that could help ESL learners to speak as a native speaker is highly desirable. Properly placing rhythmic stress is one of the important steps for teaching ESL students to have good speech production. Thus to be able to automatically detect the rhythmic stress patterns in students' speech becomes a really important functionality in this kind of computer software.

There are a number of prosodic (sometimes referred to as 'suprasegmental') features that relate to stress. Thus the perception of a syllable as stressed or unstressed may depend on its relative duration, its amplitude and its pitch. Duration is simply how long the syllable lasts. Amplitude relates to the perceived loudness of the syllable, and is a measure of its energy. Pitch is the perceptual correlate of the fundamental frequency (F_0) of the sound signal, i.e. the rate of vibration of the vocal folds during voiced segments.

A further correlate of stress is the quality of the vowel in a syllable. Vowels are split into *full* vowels and *reduced* vowels in terms of the quality based on the configuration of the tongue, jaw, and lips [1]. Full vowels tend to be more peripheral, and appear in both stressed syllables and unstressed syllables [2]. Reduced vowels, including /ə/ and /ɪ/ in New Zealand English, tend to be more central, and are only associated with unstressed syllables. Therefore, vowel quality is not a completely reliable indicator of stress [2].

In order to automatically detect rhythmic stress, prosodic features and vowel quality features as two main sets of features have been studied by many researchers using machine learning algorithms.

Waibel [3] used duration, amplitude, pitch, and spectral change to identify rhythmically stressed syllables. A Bayesian classifier, assuming multivariate Gaussian distributions, was adopted and 85.6% accuracy was achieved. Jenkin and Scordilis [4] used duration, energy, amplitude, and pitch to classify vowels into three levels of stress — primary, secondary, and unstressed. Neural networks, Markov chains, and rule-based approaches were adopted. The best overall performance was 84% by using Neural networks. Rule-based systems performed worse with 75%. Van Kuijk and Boves [5] used duration, energy, and spectral tilt to identify rhythmically stressed vowels in Dutch — a language with similar stress patterns to those of English. A simple Bayesian classifier was adopted, on the grounds that the features can be jointly modelled by a N-dimensional normal distribution. The best overall performance achieved was 68%. Our previous work [6] used duration, amplitude, pitch and vowel quality to identify rhythmically stressed vowels. Decision trees and support vector machines were applied and the best accuracy, 85%, was achieved by support vector machines.

However, the accuracies of the automatic stress detection from the literature are not high enough to be useful for a commercial system. The automatic rhythmic stress detection remains a challenge to speech recognition.

Genetic programming (GP) has grown very rapidly and has been studied widely in many areas since the early 1990s. Conrads et al. [7] demonstrated that GP could find programs that were able to discriminate certain spoken vowels and consonants without pre-processing speech signals. However, there are only a few studies using GP in the automatic speech recognition and analysis area. Most current research on automatic rhythmic stress detection uses other machine learning algorithms rather than GP.

1.1 Goals

This paper aims to use GP to develop an approach to automatic rhythmic stress detection in spoken New Zealand (NZ) English. The approach will be examined and compared with other machine learning techniques such as decision tress

(DT) and support vector machines (SVM) on a set of NZ English utterances. Specifically, we investigate:

- how GP can be used to construct an automatic rhythmic stress detector,
- whether GP outperforms DT and SVM on the automatic problem, and
- whether GP has a stronger capability of handling irrelevant features than DT or SVM.

The remainder of the paper is organised as follows: section 2 describes the GP approach; section 3 presents the experiment design; section 4 provides experiment results, and section 5 draws conclusions and discusses possible future work.

2 GP Adapted to Stress Detection

A linear-structured GP system [8] is adopted to construct an automatic rhythmic stress detector in this study. This section addresses: 1) the feature extraction and normalisation; 2) the terminal sets; 3) the function set; 4) the fitness function; and 5) the genetic parameters and termination criteria.

2.1 Feature Extraction and Normalisation

As prosodic features and vowel quality features are recognised as the two main sets of features for automatic stress detection, we also use both of them in the approach. For each of the prosodic parameters (duration, amplitude, pitch), there are many alternative measurements that can be extracted, and also many ways of normalising the features in order to reduce variation due to differences between speakers, recording situations or utterance contexts. Vowel quality features are more difficult to extract. The subsections below describe the details of the feature extraction and normalisation.

Duration Features. The absolute duration of a vowel segment is easily calculated directly from the hand labelled utterances since the start and end points of the vowel segment are clearly marked. Three different levels of normalisation are applied to the directly calculated absolute duration of a vowel segment. The first level normalisation aims to reduce the impact of the different speech rate of speakers. The second level normalisation aims to reduce the effects of the intrinsic duration properties of the vowel. Both narrow and broad methods are considered. The narrow method is to normalise the vowel segment duration by the average duration for that vowel type, as measured in the training data set. The broad method is to cluster the 20 vowel types into three categories (short vowel, long vowel and diphthong) and to normalise vowel segment durations by the average duration of all vowels in the relevant category. The third level normalisation aims to reduce the effects of the local fluctuations in speech rate within the utterance. Based on the three levels of normalisation, we have five duration features for each vowel segment:

- D_1*:* the absolute duration normalised by the length of the utterance.
- D_2*:* D_1 further normalised by the average duration of the vowel type.

- *D_3: D_1* further normalised by the average duration of the vowel category.
- *D_4: D_2* further normalised by a weighted average duration of the immediately surrounding vowel segments.
- *D_5: D_3* further normalised by a weighted average duration of the immediately surrounding vowel segments.

Amplitude Features. The amplitude of a vowel segment can be measured from the speech signal, but since amplitude changes during the vowel, there are a number of possible measurements that could be made — maximum amplitude, initial amplitude, change in amplitude, *etc.* A measure commonly understood to be a close correlate to the perception of amplitude differences between vowels is the root mean square (RMS) of the amplitude values across the entire vowel. This is the measure chosen as the basis of our amplitude features. Two levels of normalisations are applied to the RMS amplitude value across a vowel segment. The first level normalisation aims to reduce the effects of speaker voice volume differences and recording condition differences. It is done by normalising the RMS amplitude of each vowel segment against the overall RMS amplitude of the entire utterance. The second level normalisation aims to reduce the effects of changes in amplitude across the utterance. We normalise the vowel amplitude against a weighted average amplitude of the immediately surrounding vowel segments.

- *A_1:* the RMS amplitude of each vowel segment normalised by the overall RMS amplitude of the entire utterance.
- *A_2: A_1* further normalised by a weighted average amplitude of the immediately surrounding vowel segments.

Pitch Features. Pitch is calculated by measuring F_0 of the speech signal. Five pitch features of a vowel segment are computed, including the mean pitch value of the vowel segment, the pitch values at the start and at the end points of the vowel segment, and the minimum and maximum pitch values of the vowel segment. In order to reduce the effects of speaker differences caused by their different physiologies, we normalise the five pitch features of a vowel segment over the mean pitch of the entire utterance. In addition, based on the five normalised pitch features, we compute five other features that are intended to capture pitch changes over the vowel segment.

- *P_1:* the mean pitch value of the vowel normalised by the mean pitch of the utterance.
- *P_2:* the pitch value at the start point of the vowel normalised by the mean pitch of the utterance.
- *P_3:* the pitch value at the end point of the vowel normalised by the mean pitch of the utterance.
- *P_4:* the maximum pitch value of the vowel normalised by the mean pitch of the utterance.
- *P_5:* the minimum pitch value of the vowel normalised by the mean pitch of the utterance.

– P_6*:* the difference between the normalised maximum and minimum pitch values — a negative value indicates a falling pitch and a positive value indicates a rising pitch.
– P_7*:* the magnitude of P_6, which is always positive.
– P_8*:* the sign of P_6 — 1 if the pitch "rises" over the vowel segment, -1 if it "falls", and 0 if it is "flat".
– P_9*:* a boolean attribute — 1 if the pitch value at either the start point or the end point of the vowel segment cannot be detected, otherwise -1.
– P_{10}*:* a boolean attribute — 1 if the vowel segment is too short to compute meaningful mean, minimum, or maximum values, otherwise -1.

Vowel Quality Features. Since there is some flexibility in the formation of a vowel, there will in fact be a range of articulator parameter values that correspond to the same vowel. Therefore, vowel quality features are more difficult to extract. Pre-trained Hidden Markov Models (HMMs) phoneme models are used to analyse vowel segments and extract measures of vowel quality [9]. The algorithm is illustrated in figure 1 and outlined below.

Step 1. Extract vowel segments from the hand labelled utterance.
Step 2. Encode each vowel into a sequence of acoustic parameter vectors, using a 15ms Hamming window with a step size (frame period) of 11ms. These parameters consist of 12 MFCC features and the 0'th cepstral coefficient with their first and second order derivatives.
Step 3. Feed the parameter vector sequence into the 20 pre-trained HMM vowel recognisers to obtain 20 normalised acoustic likelihood scores. Each

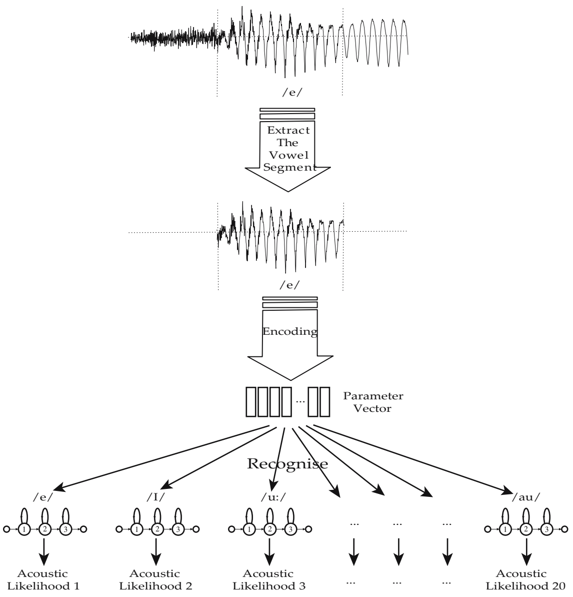


Fig. 1. Vowel quality features processing

score is the geometric mean of the acoustic likelihoods of all frames in the segment, as computed by the HMM recogniser. The scores are likelihoods that reflect how well the segment matches the vowel type of the HMM.

Step 4. Find the score of the labelled vowel type S_e, the maximum score of any full vowel phoneme S_f and the maximum score of any reduced vowel phoneme S_r from the above 20 scores.

Step 5. We then compare the scores of the best matching full vowel and the best matching reduced vowel to the score of the labelled vowel. We compute four features, two of which measure the difference between the likelihoods, and two measure the ratio of the likelihoods. In each case, we take logarithms to reduce the spread of values:

$$R_d = \begin{cases} -log(S_r - S_e) & \text{if } S_e < S_r \\ 0 & \text{if } S_e = S_r \\ log(S_e - S_r) & \text{if } S_e > S_r \end{cases} \tag{1}$$

$$F_d = \begin{cases} -\log(S_f - S_e) & \text{if } S_e < S_f \\ 0 & \text{if } S_e = S_f \\ \log(S_e - S_f) & \text{if } S_e > S_f \end{cases} \tag{2}$$

$$R_r = \log(S_e/S_r) = \log S_e - \log S_r \tag{3}$$

$$F_r = \log(S_e/S_f) = \log S_e - \log S_f \tag{4}$$

Step 6. We also compute a boolean vowel quality feature, T, to deal with cases where the vowel segment is so short that F or R cannot be calculated. If the vowel segment is less than 33ms, which is the minimum segment duration requirement of the HMM recognisers, then the value of this attribute will be 1. Otherwise, -1. If this value is 1, we set F and R to 0.

2.2 Terminal Sets

All the features introduced in section 2.1 were organised into three terminal sets. Terminal set I consists of 17 prosodic features (five duration features, two amplitude features, and 10 pitch features). Terminal set II consists of five vowel quality features. Terminal set III consists of all features combined from the prosodic and vowel quality features. Features whose values have not been explicitly mentioned in previous sections are floating point numbers with precisions up to seven digits. In each terminal set, we also include real-valued constants in the range $[-1.0, 1.0]$.

2.3 Functions

The function set contains not only the four standard arithmetic functions, but also several other arithmetic and trigonometric functions and conditional functions, as shown in equation 5.

$$FuncSet = \{abs, sqrt, cos, sin, +, -, *, /, iflt, ifpr, ifnr\} \tag{5}$$

Each of the first four mathematical operators takes a single argument. The *abs* function returns the absolute value of the argument. The protected *sqrt* function returns the square root of the argument. The *cos* or *sin* functions return the cosine or sine values of the argument respectively. Each of the $+, -, *$, and / operators takes two arguments. They have their usual meanings except that the / operator is protected division that returns 0 if its second argument is 0. The three conditional functions each takes two arguments. The *iflt* function returns 1 if the first argument is less than the second one, otherwise 0. The *ifpr* function returns the second argument if the first argument is positive, otherwise does nothing. The *ifnr* function returns the second argument if the first argument is negative, otherwise does nothing. Note that there is a redundancy in that the conditional functions could be expressed in terms of each other. There is a trade off between the increased breadth of the search space resulting from having redundant functions, and the more complex programs (hence a deeper search of the search space) resulting from a minimal set of non-redundant functions. We believe that the smaller programs that are possible with the expanded function set more than compensates for the broader search space.

2.4 Fitness Function

Error rate is used as the fitness function to evaluate programs. The classification error rate of a program is the fraction of fitness cases in the training data set that are incorrectly classified by the program. Rhythmic stressed vowel segments and unstressed vowel segments are both treated as important so that neither class is weighted over the other. In our data set, class *stressed* is represented by 1 and class *unstressed* is represented by -1. If a program's output is greater than or equal to 0, then the output is counted as a class *stressed* output. Otherwise, it is counted as a class *unstressed* output.

2.5 Parameters and Termination Criteria

In this GP system the learning process uses the tournament selection mechanism with size four and the crossover, mutation and reproduction operators. It is worth noting that in this GP system, the crossover and mutation operators are independent in that the mutation operator can be applied regardless of whether a tournament winner has also been selected for crossover, so that the sum of the crossover rate and the mutation rate may be more than 100%. The selection of parameter values used in this study is shown in Table 1. These values were obtained through prior empirical research. The unusually high mutation rates were found to be the most helpful for this problem.

The learning/evolutionary process is terminated when either of the following criteria is met:

- The classification problem has been solved on the training data set, that is, all vowel segments in the training set have been correctly classified, with the fitness of the best program being zero.

Table 1. Parameters used for GP training for three terminal sets

Parameter Kind	Parameter Name	I	II	III
Search Parameters	Population size	1024	1024	1024
	max-gwi	200	200	200
Genetic Parameters	Crossover rate	71%	57%	47%
	Mutation rate	97%	87%	83%
Program Parameters	Initial program size	80	80	80
	Max program size	256	256	256

– The number of generations reaches the pre-defined maximum number of generations without improvement (*max-gwi*). In this study, max-gwi is set at 200, which means that, if fitness values have had no improvement within 200 generations, the learning process will terminate.

3 Experiment Design

The system uses a data set collected by the School of Linguistics and Applied Language Studies at Victoria University of Wellington. The date set contains 60 utterances of ten distinct English sentences produced by six female adult NZ speakers, as part of the NZ Spoken English Database (`www.vuw.ac.nz/lals/nzsed`). The utterances were hand labelled at the phoneme level, including the time stamps of the start and the end of a phoneme segment and the phoneme label. Further, each vowel was labelled as rhythmic *stressed* or *unstressed*. There were 703 vowels in the utterances, of which 340 are marked as stressed and 363 are marked unstressed. Prosodic features and vowel quality features of each vowel segment are calculated from the hand labelled utterances.

Three experiments were conducted on the three terminal sets respectively. For each terminal set, since the data set was relatively small, a 10-fold cross validation method for training and testing the automatic rhythmic stress detectors was applied. In addition, the training and testing process was repeated ten times, that is, 100 runs of training and testing procedures were made in total for each terminal set. The average classification accuracy of the best program in each experiment is calculated from the outputs of the 100 runs.

In addition, we investigate whether scaling the feature values in the three terminal sets to the range $[-1, 1]$ results in better performance.

We also compare our GP approach with the C4.5 [10] decision tree (DT) system and a SVM system (LIBSVM [11]) on the same set of data. The SVM system uses an RBF kernel and a C parameter of 1.0.

4 Results and Analysis

4.1 Detection Performance

Terminal Set I. Table 2 shows system performance of Terminal Set I. Based on the average of 100 runs, GP achieved the best accuracy on the test set (91.9%).

The accuracy of GP was 11.5% and 12.2% higher than that of DT and SVM respectively on unscaled data, and was 11.0% and 8.4% higher on scaled. There is little evidence showing any impact of using scaled data on GP and DT. However, there is an improvement of 3.5% by using scaled data for SVM.

Table 2. Accuracy(%) for the terminal set I

	GP	DT	SVM
Unscaled	91.9	80.4	79.7
Scaled	91.6	80.6	83.2

Terminal Set II. Table 3 shows the experiment results of Terminal Set II. GP also achieved the best accuracy of 85.4%. The accuracy of GP was 5.7% and 6.3% higher than that of DT and SVM respectively on unscaled data, and was 5.7% and 4.1% higher on scaled. There is also little evidence to show any impact of scaled data.

Terminal Set III. Table 4 shows the results of Terminal Set III, which combines all the features used in Terminal Set I and Terminal Set II. Again, the best accuracy of 92.6% was achieved by GP. GP outperformed DT and SVM by 12.1% and 10.8% on unscaled data respectively, and by 12.6% and 10.6% on scaled. For all three systems, accuracies on scaled data were invariably higher than those on the unscaled data but the differences were very small.

Comparing the results of all three terminal sets, we obtained the following observations.

- On all terminal sets, regardless of whether the data are scaled or unscaled, accuracy of GP is consistently and significantly higher than that of DT and SVM. This indicates that GP is more effective than DT and SVM on the automatic stress detection problem on our data set.
- For all systems, Terminal Set I consistently returns higher accuracies than terminal set II. This indicates that either prosodic features are more accurate than vowel quality features, or that vowel quality feature extraction needs to be further improved.
- Maximising the coverage of features (using Terminal Set III) resulted in some improvement for GP, but not for DT and SVM. Since terminal set III has the most complete set of features, it is likely that not all of them are necessary in detecting stress. Therefore the difference in performance of Terminal Set I

Table 3. Accuracy(%) for the terminal set II

	GP	DT	SVM
Unscaled	85.4	79.7	79.1
Scaled	84.6	78.9	80.5

Table 4. Accuracy(%) for the terminal set III

	GP	DT	SVM
Unscaled	92.0	79.9	81.3
Scaled	92.6	80.1	82.0

and Terminal Set III could be used as an indication of how robust a system is at handling unnecessary and irrelevant features. Except for GP, both DT's and SVM's best accuracy scores dropped on Terminal Set III, therefore GP is the most robust algorithm among the three at handling unnecessary and irrelevant features on our data set.

4.2 Feature Impact Analysis

The top thirty programs in each run were analysed and the average *impact* of each terminal input in programs was computed as a percentage, as shown in Tables 5 and 6. The impact of a terminal input refers to the change of the performance of a program if all occurrences of the terminal input are removed from the program.

Table 5 shows the impact of the prosodic features. The patterns of the impact of prosodic features are similar on both unscaled data and scaled data. Three broad bands of impact emerged as high (above 5%, including all duration

Table 5. Impact analysis for prosodic features

Unscaled		Scaled	
Input	Average Impact(%)	Input	Average Impact(%)
D_5	27.7	D_5	28.2
D_3	27.4	D_3	16.5
D_2	15.1	D_4	13.4
D_4	13.7	D_1	12.9
D_1	8.3	D_2	7.8
A_1	2.3	A_2	1.4
A_2	1.1	A_1	1.4
P_2	1.1	P_5	0.6
P_1	0.7	P_3	0.6
P_3	0.7	P_2	0.6
P_8	0.6	P_4	0.5
P_4	0.4	P_1	0.4
P_5	0.3	P_8	0.4
P_{10}	0.3	P_7	0.2
P_7	0.3	P_{10}	0.2
P_6	0.1	P_9	0.2
P_9	0.1	P_6	0.1

Table 6. Impact analysis for vowel quality features

Unscaled		Scaled	
Input	Average Impact(%)	Input	Average Impact(%)
R_r	31.9	R_d	21.0
R_d	20.7	R_r	19.9
F_r	2.8	T	4.5
T	1.5	F_d	0.76
F_d	0.34	F_r	0.38

features), medium (1% to 5%, including amplitude features), and low (under 1% including all pictch features), corrsponding exactly with the three feature categories - duration, amplitude and pitch. This indicates duration has a bigger impact than amplitude while pitch has the smallest impact. On both unscaled and scaled data set, D_5 and D_3 are ranked as the first and second, indicating that normalisations of a duration feature over the average duration of a vowel category is better than that over the average duration of a vowel type.

The ranking of duration, amplitude and pitch in terms of impact in this study matches the result in [6]. However, only one experiment was conducted in this study whereas seven experiments with various combinations of the feature sets were conducted in [6], where DT and SVM were used. This suggests that: 1) GP has stronger feature selection ability than DT and SVM on the problem; 2) GP can automatically handle a large number of features; and 3) GP can automatically select features that are only important to a particular domain.

As shown in Table 6, R_d and R_r have a much larger impact than F_d, F_r, and T on both unscaled and scaled data. On unscaled data R_d's impact (31.9%) is larger than R_r's impact (20.7%), whereas on scaled data the two features display a similar impact. The results suggest that the reduced vowel quality features are far more useful than full vowel quality features, regardless of whether differences or ratios are used.

5 Conclusions and Future Work

The goal of this paper was to develop an approach to using GP for automatic rhythmic stress detection in spoken NZ English. A range of prosodic and vowel quality features were calculated, normalised and/or scaled from vowel segments in speech. The approach was tested on 60 female adult utterances. A maximum average accuracy of 92.61% was achieved by our GP system.

The results strongly support the use of GP to construct a more effective automatic rhythmic stress detector than DT and SVM. Furthermore, according to our data set, GP is more robust at handling large numbers of unnecessary features and maintaining high performance than DT and SVM. GP also has a stronger automatic feature selection ability than DT and SVM.

In addition, prosodic features appear to be more accurate in detecting stress than vowel quality features, duration features being specially identified as the most important features. If using vowel quality, reduced vowel quality features are more useful than full vowel quality features.

In future work, we will further analyse the GP programs to understand the specific relationship amongst the feature terminals and the perceived stressed and unstressed vowels in order to determine whether the generated GP program with/without adapting can be applied to any other kind of data sets. We are also planning to investigate the possibility of having GP automatically perform higher level normalisations of the prosodic features and calculate vowel quality features directly from acoustic likelihoods in order to erase the limitation of the manual pre-process of the features.

Acknowledgment

The data set used in the paper was provided by Dr Paul Warren of the School of Linguistics and Applied Language Studies at Victoria University of Wellington.

References

1. Ladefoged, P.: Three Areas of experimental phonetics. Oxford University Press, London (1967)
2. Ladefoged, P.: A Course in Phonetics. third edn. Harcourt Brace Jovanovich, New York (1993)
3. Waibel, A.: Recognition of lexical stress in a continuous speech system - a pattern recognition approach. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Tokyo, Japan (1986) 2287–2290
4. Jenkin, K.L., Scordilis, M.S.: Development and comparison of three syllable stress classifiers. In: Proceedings of the International Conference on Spoken Language Processing, Philadelphia, USA (1996) 733–736
5. van Kuijk, D., Boves, L.: Acoustic characteristics of lexical stress in continuous speech. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. Volume 3., Munich, Germany (1999) 1655–1658
6. Xie, H., Andreae, P., Zhang, M., Warren, P.: Detecting stress in spoken English using decision trees and support vector machines. Australian Computer Science Communications (Data Mining, CRPIT 32) **26** (2004) 145–150
7. Conrads, M., Nordin, P., Banzhaf, W.: Speech sound discrimination with genetic programming. In: Proceedings of the First European Workshop on Genetic Programming. (1998) 113–129
8. Francone, F.D.: Discipulus owner's manual (2004)
9. Xie, H., Andreae, P., Zhang, M., Warren, P.: Learning models for English speech recognition. Australian Computer Science Communications (Computer Science, CRPIT 26) **26** (2004) 323–330
10. Quinlan, J.: C4.5: Programs for machine learning. Morgan Kaufmann (1993)
11. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf (2003)
12. Koza, J.R.: Genetic Programming — On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
13. Dy, J.G., Brodley, C.E.: Feature selection for unsupervised learning. Journal of Machine Learning Research **5** (2004) 845–889

Localisation Fitness in GP for Object Detection

Mengjie Zhang and Malcolm Lett

School of Mathematics, Statistics and Computer Science,
Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
{mengjie, malcolm}@mcs.vuw.ac.nz

Abstract. This paper describes two new fitness functions in genetic programming for object detection particularly object localisation problems. Both fitness functions use weighted F-measure of a genetic program and consider the localisation fitness values of the detected object locations, which are the relative weights of these locations to the target object centers. The first fitness function calculates the weighted localisation fitness of each detected object, then uses these localisation fitness values of all the detected objects to construct the final fitness of a genetic program. The second fitness function calculates the average locations of all the detected object centres then calculates the weighted localisation fitness value of the averaged position. The two fitness functions are examined and compared with an existing fitness function on three object detection problems of increasing difficulty. The results suggest that almost all the objects of interest in the large images can be successfully detected by all the three fitness functions, but the two new fitness functions can result in far fewer false alarms and spend much less training time.

1 Introduction

As more and more images are captured in electronic form, the need for programs which can detect objects of interest in a database of images is increasing. For example, it may be necessary to detect all tumors in a database of x-ray images, find all cyclones in a database of satellite images, detect a particular face in a database of photographs, or detect all tanks, trucks or helicopters in a set of images [1, 2, 3, 4]. This field is typically called *object detection.*

Object detection is an important field of research in image analysis and processing for many modern tasks. It is the process of finding locations of the objects of interest within images, known as *object localisation*, and determining the types or classes of the objects found, known as *object classification.* Object localisation is sometimes referred to object detection, as it also involves classifying all the objects of interest from background. It is noted that object detection is a difficult task, particularly when the objects of interest are irregular and/or the background is highly cluttered.

Genetic programming (GP) is a relatively recent and fast developing approach to automatic programming [5, 6, 7]. In GP, solutions to a problem can be represented in different forms but are usually interpreted as computer programs. Darwinian principles of natural selection and recombination are used to evolve a population of programs towards an effective solution to specific problems. The flexibility and expressiveness of computer program representation, combined with the powerful capabilities of evolutionary search, make GP an exciting new method to solve a great variety of problems. A strength of this approach is that evolved programs can be much more flexible than the highly constrained, parameterised models used in other techniques such as neural networks and support vector machines. GP has been applied to a range of object recognition tasks such as shape classification, face identification, and medical diagnosis with some success [5, 8, 9, 10, 11].

Finding a good fitness function for a particular object detection problem is an important but difficult task in developing a GP system. Various fitness functions have been devised for object detection, with varying success [5, 9, 11, 12, 13]. These tend to combine many parameters using scaling factors which specify the relative importance of each parameter, with no obvious indication of what scaling factors are good for a given problem. Many of these fitness functions for localisation require clustering to be performed to group multiple localisations of single objects into a single point before the fitness is determined [14, 13, 12]. Other measures are then incorporated in order to include information about the pre-clustered results (such as how many points have been found for each object). While some of these systems achieved good detection rates, many of them resulted in a large number of false alarms. In particular, the clustering process during the evolutionary process made the training time very long.

This paper aims to investigate two new fitness functions in GP for object detection, in particular localisation, with the goal of improving the detection performance particularly reducing false alarms and improving the evolutionary training efficiency. The two fitness functions will be examined and compared with an existing fitness function on a sequence of object detection problems of increasing difficulty.

The remainder of this paper is organised as follows. Section 2 describes the basics of object detection. Section 3 describes the overview of our GP approach. Section 4 details the two new fitness functions with design requirements. Section 5 describes experiment design and configurations and section 6 presents the results with discussions. Finally, we draw conclusions in section 7.

2 Object Detection

The process for object detection is shown in Figure 1. A raw image is taken and a trained localiser applied to it, producing a set of points found to be the positions of these objects. Single objects could have multiple positions ("localisations") found for them, however ideally there would be exactly one localisation per object. Regions of the image are then "cut out" at each of the positions specified. Each of these cutouts are then classified using the trained classifier.

This method treats all objects of multiple classes as a single "object of interest" class for the purpose of localisation, and the classification stage handles attaching correct class labels. Compared with the single-stage approach [10, 11],

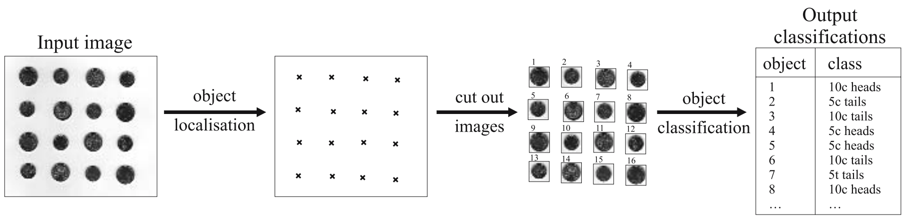


Fig. 1. Object Detection Process

this has the advantage that the training is easier for both stages as a specific goal is focused on the training of each of the two stages. The first is tailored to achieving results as close to the object centres as possible (to achieve high "positional accuracy"), while the second is tailored to making all classifications correct (high "classification accuracy").

The accuracy of the positions found by the localiser is important, as incorrect results from this first stage will need to be handled by the second stage. Any false alarms from the localisation stage could cause problems with the classification stage, unless it is able to classify these as "background". If the positions found are not close enough to the object centres then the classification stage will likely not handle it well.

The object localisation stage is performed by means of a window which sweeps over the whole image, and for each position extracts the features and passes them to the trained localiser. The localiser then determines whether each position is an object or not (i.e. background).

3 Genetic Programming Applied to Object Detection

Our work will focus on object localisation using genetic programming. Figure 2 shows an overview of this approach, which has a learning process and a testing procedure. In the learning/evolutionary process, the evolved genetic programs use a square input field which is large enough to contain each of the objects of interest. The programs are applied at many sampled positions within the images in the training set to detect the objects of interest. If the program localiser returns a value greater than or equal to zero, then this position is considered the centre of an object of interest; otherwise it is considered background. In the test procedure, the best evolved genetic program obtained in the learning process is then applied, in a moving window fashion, to the whole images in the test set to measure object detection performance.

In this system, we used tree structures to represent genetic programs [6]. The ramped half-and-half method [5] was used for generating programs in the initial population and for the mutation operator. The proportional selection mechanism and the reproduction, crossover and mutation operators [5] were used in the learning process.

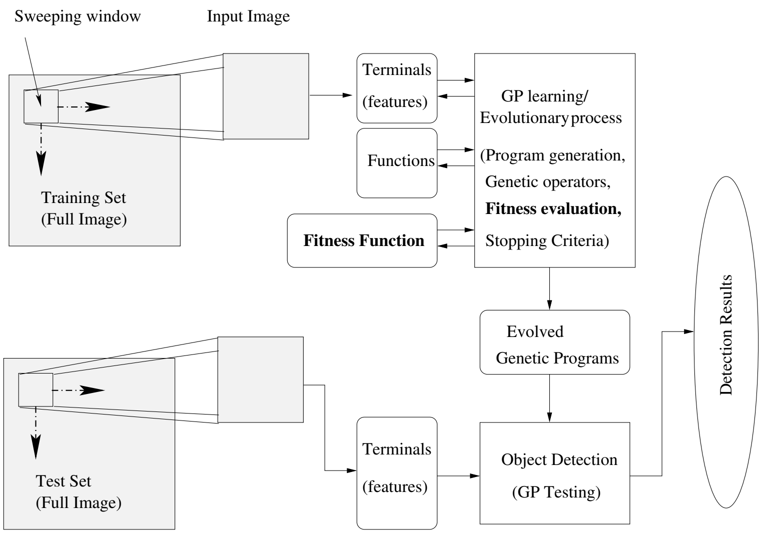
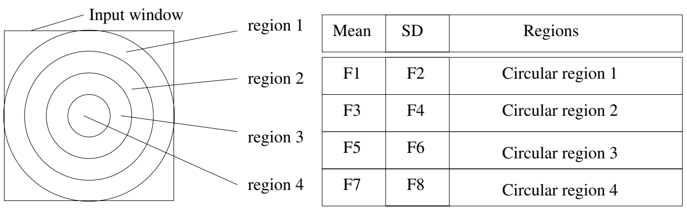

Sweeping window
Input Image
Training Set (Full Image)
Terminals (features)
Functions
Fitness Function
GP learning/ Evolutionary process
(Program generation, Genetic operators, Fitness evaluation, Stopping Criteria)
Evolved Genetic Programs
Detection Results
Test Set (Full Image)
Terminals (features)
Object Detection (GP Testing)

Fig. 2. An overview of the GP approach for object detection

3.1 Terminal Set

For object detection problems, terminals generally correspond to image features. In this approach, the features are extracted by calculating the mean and standard deviation of pixel values within each of the circular regions, as shown in Figure 3. This set of features has the advantages of being rotationally invariance. In addition, we also used a constant terminal.



Mean	SD	Regions
F1	F2	Circular region 1
F3	F4	Circular region 2
F5	F6	Circular region 3
F7	F8	Circular region 4

Fig. 3. Eight concentric circular region features

3.2 Function Set

In the function set, the four standard arithmetic and a conditional operation was used to form the non-terminal nodes:

$$FuncSet = \{+, -, *, /, if\}$$

The $+$, $-$, and $*$ operators have their usual meanings — addition, subtraction and multiplication, while / represents "protected" division which is the usual division operator except that a divide by zero gives a result of zero. Each of these functions takes two arguments. The *if* function takes three arguments. The first argument, which can be any expression, constitutes the condition. If the first argument is positive, the *if* function returns its second argument; otherwise, it returns its third argument. The *if* function allows a program to contain a different expression in different regions of the feature space, and allows discontinuous programs, rather than insisting on smooth functions.

Construction of the fitness functions is the main focus of this paper, which will be described in the next section.

4 Fitness Functions

4.1 Design Considerations

During the evolutionary process for object detection, we expect that the evolved genetic programs only detect the objects when the sweeping window is centred over these objects. However, in the usual case (except the ideal case), these evolved genetic programs will also detect some “objects” not only when the sweeping window is within a few pixels of the centre of the target objects, but also when the sweeping window is centred over a number of cluttered pieces of background. Clearly, these “objects” are not those we expected but are false alarms.

Different evolved genetic programs typically result in different numbers of false alarms and such differences should be reflected when these programs are evaluated by the fitness function.

When designing a fitness function for object detection problems, a number of considerations need to be taken into account. At least the following requirements should be considered.

Requirement 1. The fitness function should encourage a greater number of objects to be detected. In the ideal case, all the objects of interest in large images can be detected.

Requirement 2. The fitness function should prefer a fewer number of false alarms on the background.

Requirement 3. The fitness function should encourage genetic programs to produce detected object positions closer to the centres of the target objects.

Requirement 4. For a single object to be detected, the fitness function should encourage programs to produce fewer detected “objects” (positions) within a few pixels from the target center.

Requirement 5. For two programs which produce the same number of detected “objects” for a single target object but the “objects” detected by the first program are closer to the target object centre than those detected by the second program, the fitness function should rank the first program better than the second.

Some typical examples of these requirements are shown in figure 4. In this figure, the circles are target objects and squares are large images or regions. A cross (x) represents a detected object. In each of the five cases, the left figure is associated with a better genetic program than the right.

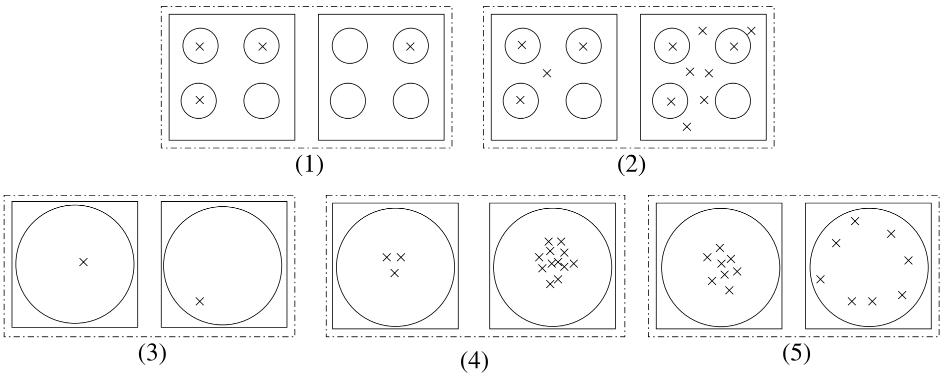


Fig. 4. Examples of the design considerations of the fitness function

4.2 An Existing Fitness Function

As the goal is to detect the target objects with no or a small number of false alarms, many GP systems uses a combination of detection rate and false alarm rate or recall and precision as the fitness function. For example, a previous GP system uses the following fitness function [10]:

$$fitness_{CBF} = A \cdot (1 - DR) + B \cdot FAR + C \cdot FAA \tag{1}$$

where DR, FAR, and FAA are detection rate (the number of small objects correctly reported by a detection system as a percentage of the total number of actual objects in the images), false alarm rate (also called *false alarms per object*, the number of non-objects incorrectly reported as objects by a detection system as a percentage of the total number of actual objects in the images), and false alarm area (the number of false alarm pixels which are not object centres but are incorrectly reported as object centres before clustering), respectively, and A, B, C are constant weights which reflect the relative importance of detection rate versus false alarm rate versus false alarm area.

Basically, this fitness function has considered requirement 1, and partially considered requirements 2 and 4, but does not take into accounts of requirements 3 and 5. Although this fitness function performed reasonably well on some problems, it still produced many false alarms and the evolutionary training time was still very long [10]. Since this method used clustering before calculating the fitness, we refer to it as *clustering based fitness*, or CBF for short.

4.3 First New Fitness Function — LFWF

To avoid a very large false alarm rate (greater than 100% for difficult problems) in the training process, we use precision and recall, both of which have the range between [0, 1], to construct the new fitness functions. Precision refers to the number of objects correctly localised/detected by a GP system as a percentage of the total number of object localised/detected by the system. Recall refers to the number of objects correctly localised by a system as a percentage of total number of target objects in a data set. Note that precision/recall and detection rate/false alarm rate have internal relationship, where the value of one pair for a problem can be calculated from the other for the same problem.

During the object localisation process, a genetic program might consider many pixel positions in an image as object centres, particularly for those pixels surrounding the target object centres. For presentation convenience, we call each object centre localised in an image by a genetic program a *localisation*.

Unlike the previous fitness function CBF, our first fitness function is based on a "Localisation Fitness Weighted F-measure" (LFWF), which attempts to acknowledge the worth/goodness of individual localisations made by the genetic program. Instead of using either correct or incorrect to represent a localisation, each localisation is allocated a weight (referred to as the *localisation fitness*) which represents its individual worth and counts towards the overall fitness. If all localisations have a high localisation fitness, then the overall fitness of this genetic program will be good.

Each weight is calculated based on the distance of the localisation from the centre of the closest object, as shown in Equation 2.

$$\text{localisationFitness}(x, y) = \begin{cases} 1 - \frac{\sqrt{x^2+y^2}}{r} & \text{, if } \sqrt{x^2+y^2} \le r \\ 0 & \text{, otherwise} \end{cases} \tag{2}$$

where $\sqrt{x^2+y^2}$ is the distance of the localisation position (x, y) from target object centre, and r is called the "localisation fitness radius", defined by the user. In this system, r is set to a half of the square size of the input window, which is also the radius of the largest object.

In order to deal with all the situations in the five design requirements, we used the localisation fitness to construct our first new fitness function, as shown in Equations 3 to 5. The precision and recall are calculated by taking the localisation fitness for all the localisations of each object and dividing this by the total number of localisations or total number of target objects respectively.

$$\text{WP}_1 = \frac{\sum_{i=1}^{N} \sum_{j=1}^{L_i} \text{localisationFitness}(x_{ij}, y_{ij})}{\sum_{i=1}^{N} L_i} \tag{3}$$

$$\text{WR}_1 = \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{L_i} \text{localisationFitness}(x_{ij}, y_{ij})}{L_i}}{N} \tag{4}$$

$$\text{fitness}_{LFWF} = \frac{2 \times \text{WP}_1 \times \text{WR}_1}{\text{WP}_1 + \text{WR}_1} \tag{5}$$

where N is the total number of target objects, (x_{ij}, y_{ij}) is the position of the j-th localisation of object i, L_i is number of localisations made to object i, WP_1 and WR_1 are the weighted precision and recall, and $\textit{fitness}_{LFWF}$ is the localisation fitness weighted F-measure, which is used as the first new fitness function.

Main Characteristics. The fitness function has a number of properties. Firstly, the main parameter in this fitness function is the *localisation fitness*, which can by easily determined in the way presented here. This has an advantage over the existing methods which have many parameters whose values usually need to be manually determined. Secondly, in the previous approaches, the multiple localisations of each object must be clustered into one group and its centre found. While this is not a too difficult task, it is very time consuming to do during training. This new fitness function does not require clustering before the fitness is calculated. We expect that the new fitness function will do a better job in terms of reducing false alarms and evolutionary training time.

4.4 Second New Fitness Function — APWF

Our second fitness function is designed by combining the idea behind the clustering based fitness function into our first new weighted fitness function. Rather than using the averaged localisation fitness values for all localisations for each object in LFWF, we calculate the average positions of all the localisations for each object first then use the localisation fitness of the averaged position for all localisations to calculate the weighted precision and recall, as shown in equations 6, 7, and 8. We refer to this fitness function as *APWF*, as it is based on the "average position weighted F-measure".

$$\mathrm{WP}_2 = \frac{\sum_{i=1}^{N} \mathrm{localisationFitness}(\frac{\sum_{j=1}^{L_i} x_{ij}}{L_i}, \frac{\sum_{j=1}^{L_i} y_{ij}}{L_i}) \cdot L_i}{\sum_{i=1}^{N} L_i} \tag{6}$$

$$\mathrm{WR}_2 = \frac{\sum_{i=1}^{N} \mathrm{localisationFitness}(\frac{\sum_{j=1}^{L_i} x_{ij}}{L_i}, \frac{\sum_{j=1}^{L_i} y_{ij}}{L_i})}{N} \tag{7}$$

$$\mathrm{fitness}_{APWF} = \frac{2 \times \mathrm{WP}_2 \times \mathrm{WR}_2}{\mathrm{WP}_2 + \mathrm{WR}_2} \tag{8}$$

where WP_2, WR_2 and $fitness_{APWF}$ are the weighted precision, recall and fitness function in this case. This fitness function *mimics* the idea of clustering based fitness functions and we are interested in the investigation of effect of it and compare it with our first new fitness function and the clustering based fitness function.

5 Experiment Design and Configurations

We used three image data sets of New Zealand 5 and 10 cent coins in the experiments. Examples are shown in Figure 5. The data sets are intended to provide object localisation/detection problems of increasing difficulty. The first data set

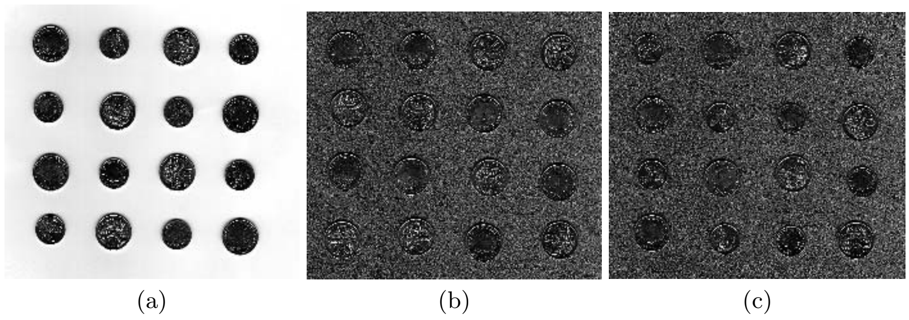


Fig. 5. Sample images in the three data sets. (a) Easy; (b) Medium difficulty; (c) Hard.

(*easy*) contains images of tails and heads of 5 and 10 cent coins against an almost uniform background. The second (*medium difficulty*) is of 10 cent coins against a noisy background, making the task harder. The third data set (*hard*) contains tails and heads of both 5 and 10 cent coins against a noisy background.

We used 24 images for each data set in our experiments and equally split them into three sets: a training set for learning good genetic programs, a validation set for monitoring the training process to avoid overfitting, and a test set to measure object detection performance.

To give a fair comparison for the three fitness functions., the "localisation recall (LR) and precision (LP)" were used to measure the final object detection accuracy on the test set. LR is the number of objects with one or more correct localisations within the localisation fitness radius at the target object centres as a percentage of the total number of target objects, and LP is the number of correct localisations which fall within the localisation radius at the target object centres as a percentage of the total number of localisations made. In addition, we also check the "Extra Localisations" (ExtraLocs) for each system to measure how many extra localisations were made for each object. The training efficiency of the systems is measured with the number of training generations and the CPU (user) time in second.

We used a population of 500 genetic programs for evolution in each experiment run. The reproduction rate, crossover rate and mutation rate were 5%, 70% and 25%, respectively. The program size was initialised to 4 and it could increase to 8 during evolution. The system run 50 generations unless it found a solution, in which case the evolution was terminated early. A total number of 100 runs were performed for each fitness function on each data set and average results are presented in the next section.

6 Results

Table 1 shows the results of the GP systems with the three fitness functions. The first line shows that, for the easy data set, the GP system with the existing fitness function, CBF, achieved an average localisation recall 99.99% and an average

Table 1. Results of the GP systems with the two fitness functions

Dataset	Fitness function	Test Accuracy			Training Efficiency	
		LR (%)	LP (%)	ExtraLocs	Generations	time(sec)
Easy	CBF	99.99	98.26	324.09	13.69	178.99
	LFWF	99.99	99.36	98.35	36.44	111.33
	APWF	99.98	99.35	123.17	38.11	113.38
Medium	CBF	99.60	83.19	804.88	36.90	431.94
	LFWF	99.90	94.42	95.69	34.35	105.56
	APWF	99.75	92.34	163.65	37.88	118.13
Hard	CBF	98.22	75.54	1484.51	31.02	493.65
	LFWF	99.53	87.65	114.86	33.27	107.18
	APWF	99.27	83.26	187.50	37.27	123.77

of localisation precision 98.26%, and resulted in 324.09 extra localisations on average. The average number of generations spent on training was 13.69 and the average training CPU user time was 178.99 seconds.

The results on the easy data set show that all the three fitness functions achieved almost perfect test accuracy. Almost all the objects of interest in this data set were successfully localised with very few false alarms (both LR and LP are very close to 100%), reflecting the fact that the detection task in this data set is relatively easy. However, the extra locations and the training time of the three approaches are quite different. Both the two new fitness functions (LFWF and APWF) produced a far fewer number of extra localisations per object than clustering based fitness function (CBF) and the gap between them is significant. Although the CBF approach used only 13.69 generations on average, which are considerably fewer than the other two approaches, it actually spent about 50% longer training time. This confirms our early hypothesis that the clustering process in the CBF approach is time consuming and the approaches with the two new fitness functions are more efficient than that with CBF. The approach with LFWF used slightly less time and produced fewer extra localisations than APWF.

The results on the other two data sets show a similar pattern in terms of the number of extra localisations and training time. The systems with LFWF and APWF always produced a significantly fewer number of extra localisations and a much short training time than CBF and the LFWF was the best over the three fitness functions. In addition, although almost all the objects of interest in the large images were successfully detected (LRs are almost 100%), the localisation precisions achieved by LFWF and APWF were significantly better than CBF, suggesting that the two new fitness functions outperform the existing one in terms of reducing false alarms. Also, LFWF outperformed APWF on these two data sets.

As expected, performance on the three data sets deteriorated as the degree of difficulty of the object detection problem was increased.

6.1 Detection Map Analysis

To give an intuitive view of detection performance of the three fitness functions, we checked the "detection maps" of some objects in the test set. Figure 6 (a), (b) and (c) show the detection maps for the same 15 objects in the medium difficulty data set produced by the three approaches. The black pixels in these maps indicate the localisations of the 15 objects produced using the three fitness functions. The "background" means that no objects were found in those positions.

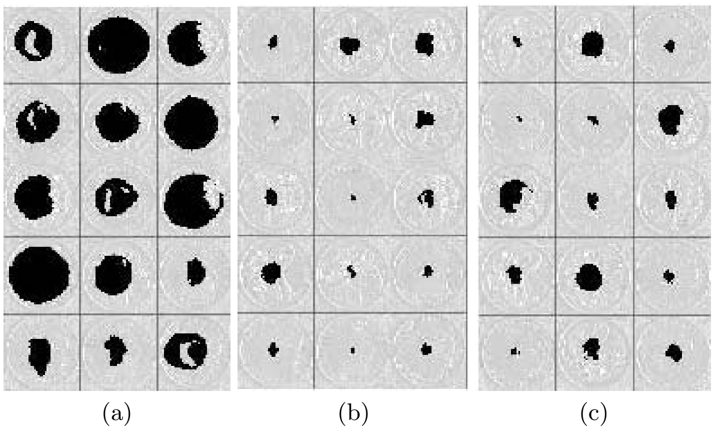


Fig. 6. Sample object detection maps. (a) CBF; (b) LFWF; (c) APWF.

As shown in the figure, the clustering based fitness function CBF resulted in a huge number of extra localisations for all the 15 objects detected. The LFWF method, however, only resulted in a small number of extra localisations. Although the APWF method produced more localisations than the LFWF method, it was much better than the CBF method in producing extra localisations. These maps confirm that the two new fitness functions were more effective than the clustering based fitness function on these problems and the LFWF performed the best.

7 Conclusions

This paper described two new fitness functions in genetic programming for object detection, particularly object localisation problems. Rather than using a clustering process to determine the number of objects detected by the GP systems, the two new fitness functions introduced a weight called localisation fitness to represent the goodness of the detected objects and used weighted F-measures. The first fitness function LFWF calculated the weighted localisation fitness of each detected object, then used these localisation fitness values of all the detected objects to construct the final fitness measure of a genetic program. The second fitness function APWF calculated the average locations of all the detected object centres then calculated the weighted localisation fitness value of the averaged position. The two fitness functions were examined and compared with an existing fitness function on three object detection problems of increasing difficulty. The results suggest that the two new fitness functions outperformed the existing clustering based fitness function in terms of both detection accuracy and training efficiency. The LFWF approach achieved the better performance than the APWF approach.

In the future, we will apply the new approach to other object detection problems including object overlapping situations. We will also investigate new ways for further reducing training time by effectively organising training examples.

References

1. Gader, P.D., Miramonti, J.R., Won, Y., Coffield, P.: Segmentation free shared weight neural networks for automatic vehicle detection. Neural Networks **8** (1995) 1457–1473
2. Roitblat, H.L., Au, W.W.L., Nachtigall, P.E., Shizumura, R., Moons, G.: Sonar recognition of targets embedded in sediment. Neural Networks **8** (1995) 1263–1273
3. Roth, M.W.: Survey of neural network technology for automatic target recognition. IEEE Transactions on neural networks **1** (1990) 28–43
4. Waxman, A.M., Seibert, M.C., Gove, A., Fay, D.A., Bernandon, A.M., Lazott, C., Steele, W.R., Cunningham, R.K.: Neural processing of targets in visible, multispectral ir and sar imagery. Neural Networks **8** (1995) 1029–1051
5. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications. Morgan Kaufmann Publishers; Heidelburg (1998)
6. Koza, J.R.: Genetic programming : on the programming of computers by means of natural selection. Cambridge, Mass. : MIT Press, London, England (1992)
7. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge, Mass. : MIT Press, London, England (1994)
8. Song, A., Ciesielski, V., Williams, H.: Texture classifiers generated by genetic programming. In Proceedings of the 2002 Congress on Evolutionary Computation, IEEE Press (2002) 243–248
9. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann (1993) 303–309
10. Zhang, M., Andreae, P., Pritchard, M.: Pixel statistics and false alarm area in genetic programming for object detection. In Applications of Evolutionary Computing, LNCS Vol. 2611, Springer-Verlag (2003) 455–466
11. Zhang, M., Ciesielski, V., Andreae, P.: A domain independent window-approach to multiclass object detection using genetic programming. EURASIP Journal on Signal Processing, **2003** (2003) 841–859
12. Smart, W., Zhang, M.: Classification strategies for image classification in genetic programming. In Proceeding of Image and Vision Computing Conference, Palmerston North, New Zealand (2003) 402–407
13. Howard, D., Roberts, S.C., Brankin, R.: Target detection in SAR imagery by genetic programming. Advances in Engineering Software **30** (1999) 303–311
14. Bhowan, U.: A domain independent approach to multi-class object detection using genetic programming. BSc Honours research project, School of Mathematical and Computing Sciences, Victoria University of Wellington (2003)

Immune Multiobjective Optimization Algorithm for Unsupervised Feature Selection

Xiangrong Zhang, Bin Lu, Shuiping Gou, and Licheng Jiao

National Key Lab for Radar Signal Processing,
Institute of Intelligent Information Processing,
Xidian University, Xi'an 710071, China
xrzhang@mail.xidian.edu.cn

Abstract. A feature selection method for unsupervised learning is proposed. Unsupervised feature selection is considered as a combination optimization problem to search for the suitable feature subset and the pertinent number of clusters by optimizing the efficient evaluation criterion for clustering and the number of features selected. Instead of combining these measures into one objective function, we make use of the multiobjective immune clonal algorithm with forgetting strategy to find the more discriminant features for clustering and the most pertinent number of clusters. The results of experiments on synthetic data and real datasets from UCI database show the effectiveness and potential of the method.

1 Introduction

Intuitively, more features can describe a given target better and are in favor of the improvement of the discriminating performance. However, it is not the case in practice. One may extract the potentially useful features for many learning domains. However, some of the features may be redundant or irrelevant, and some may even misguide the learning results. In such a case, removing these "noise" features will often lead to better performance.

Feature selection is defined as the process of choosing a subset from the original predictive variables by eliminating redundant features and those with little or no predictive information. In supervised learning, it is a popular technique and has been used in various applications because it can improve the performance of classifiers and lower or even avoid the dimension curse. According to the evaluation criterions, feature selection algorithms can be divided into filters and wrappers [1]. The former performs feature selection independently of any learning algorithm. On the other hand, the candidate feature subset is evaluated by the classification accuracy in wrappers.

Compared with supervised learning, only in recent years, some investigations on feature selection for unsupervised leaning have been made gradually. The unsupervised feature selection algorithms also can be classified into filter and wrapper mehtods. Distance entropy based algorithm [2] and the method based on feature similarity [3] are examples of filters. Most researches on unsupervised feature selection belong to wrappers. In [4], mixture models are used to implement feature selection and clustering simultaneously. In [5], a separation criterion is used for feature evaluation, and the minimum description length penalty term is added to the log-likelihood criterion to determine the number clusters. The multiobjective genetic algorithm and an evolutionary local selection algorithm are used in [6] and [7] respectively.

In this paper, an unsupervised feature selection algorithm based on wrapper frame is proposed, in which FCM algorithm is applied to form the clusters based on the selected features. Generally, a feature selection algorithm involves two sides: a feature evaluation criterion and a search algorithm. A validity measure for fuzzy clustering is applied to evaluate the quality of the features for FCM algorithm. In addition, suitable feature selection and the pertinent number of clusters must be optimized at the same time in unsupervised feature selection since different feature subsets may lead to different cluster structures. The task presents a multi-criterion optimal problem. Recently, evolutionary computation algorithms get widely applications in feature selection [8] and synthesis [9] [10] to improve the performance and reduce the feature dimension as well. In paper [11], the Immune Clonal Algorithm (ICA) is applied to search for the optimal feature subset for classification in which different objectives are combined into a single objective function. The main drawback of using ICA here lies in that it is difficult to explore different possibilities of trade-offs among objectives. Then the Immune Forgetting Multiobjective Optimization Algorithm (IFMOA) [12] is used to find a set of nondominated solutions which imply the more discriminate features and the more pertinent number of clusters.

2 Evaluation of Features for Clustering

Feature selection for clustering is to search for fewer features that best uncovers "natural" groups from data according to some criterion. So we need one or more criterions to evaluate the quality of clustering based on different feature subsets. Among them, the within-cluster scatter and between-cluster separation criterion has been widely used. Two objective functions are used to compute these measurements independently in [7]. The scatter seperability criterion $trace(S_w^{-1}S_b)$ is used in [5]. A DB index is used in [6], which is more suitable for hard cluster. Since different cluster algorithms need different measures for evaluation, a validity measure for fuzzy clustering proposed in [13] is used for the evaluation of FCM.

After implementing FCM on data set $X = \{X_i; i = 1, 2, \cdots, n\}$, we can get the centroids $Z_j\,(j = 1, 2, \cdots c)$ and the fuzzy membership μ_{ij} $(i = 1, 2, \cdots, n,\ j = 1, 2, \cdots, c)$ of vector *i* belonging to cluster *j*, in which $\mu_{ij} \in (0,1)$. Then, the fuzzy deviation of a sample X_i is defined as

$$d_{ij} = \mu_{ij} \left\| X_i - Z_j \right\| \tag{1}$$

where $\|\cdot\|$ is the usual Euclidean norm. Thus d_{ij} is just the Euclidean distance between X_i and Z_j weighted by the fuzzy membership of data *i* belonging to cluster *j*.

For each cluster *j*, the summation of the squares of fuzzy deviation of each data point is called the variation of cluster *j*, which is denoted by σ_j^2 and is defined as

$$\sigma_j^2 = \sum_{i=1}^{n} (d_{ij})^2 = \sum_{i=1}^{n} \mu_{ij}^2 \left\| X_i - Z_j \right\|^2 . \tag{2}$$

Then the total variation of data set X is

$$\sigma = \sum_{j=1}^{c} \sigma_j^2 . \tag{3}$$

Consequently, the compactness of the fuzzy c-partition of the data set is defined as the ratio of the total variation to the size of the data set σ / n . The smaller the ratio is, the more compact the clusters are. And σ_j^2 / n_j is called the compactness of cluster *j* where $n_j = \sum_{i=1}^{n} \mu_{ij}$ is the fuzzy cardinality of cluster *j*.

On the other hand, the separation of the fuzzy c-partition is defined as:

$$d_{\min} = \min_{i,j=1,\cdots,c,i\neq j} \left\| Z_i - Z_j \right\|^2 . \tag{4}$$

$d_{\min}$ measures the minimum distance between cluster centroids. Larger $d_{\min}$ indicates that all the clusters are separated.

Then, the compactness and the separation validity function or Xie-Beni index is defined as

$$XB = \frac{\sigma / n}{d_{\min}} . \tag{5}$$

A smaller XB indicates a partition in which the clusters are overall compact, and separate to each other. Thus, our goal in this paper is to find the feature subset for FCM with the smallest XB .

It is noted that, in FCM algorithm, the objective function is

$$J_q = \sum_{i=1}^{n} \sum_{j=1}^{c} \mu_{ij}^q \left\| X_i - Z_j \right\|^2 . \tag{6}$$

So, when $q = 2$ in FCM, Xie-Beni index can be calculated by $XB = J_2 / nd_{\min}$, from which, it is shown that the smallest XB implies the smallest J_2 .

3 The Need for Determining the Number of Clusters

Unsupervised feature selection is difficult because one has not effective guidance for finding relevant features without class labels. And the task is made more challenging when the number of clusters is unknown. However, different feature subsets lead to varying cluster structure. Fig.1, for example, indicts it clearly. It can be found that the numbers of clusters are 1, 2, 3 using feature subsets $\{x_1\}$, $\{x_2\}$ and $\{x_1, x_2\}$ respectively. It is unreasonable using a fixed number of clusters in the process of feature selection. Therefore, suitable feature subset and the pertinent number of clusters must be considered at the same time.

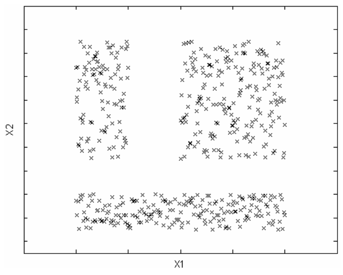


Fig. 1. The influence of feature subset to the number of clusters

In literatures, various criterions are used for finding the number of clusters, for example, Bayesian Information Criterion [14], minimum message length is used in [4], and a minimum description length penalty is used in [5]. Instead of estimating the number of clusters, *c*, we get it by encoding it as a part of the individual solution along with the feature selection. Then each individual in Immune Multiobjective Optimization Algorithm represents a feature subset and a number of clusters.

4 IFMOA for Unsupervised Feature Selection

As discussed above, more than one criterion is needed for unsupervised feature selection, which will be the objective functions for IFMOA to be optimized to find the more discriminant features and the pertinent number of clusters with fewer features.

4.1 Objective Functions

A validity measure for clustering on the given feature subset is needed first. From section 2, it is found that a smaller XB indicates a partition in which the clusters are overall compact, and discriminated from each other. So we minimize the following objective function.

$$f_1 = XB \,. \tag{7}$$

However, as Xie and Beni pointed in [13], the criterion above prefers larger number of clusters. In order to compensate for the previous criterion's bias to the number of clusters, then we minimize the second objective.

$$f_2 = c \tag{8}$$

where c is the number of selected clusters.

The third objective is to find the feature subset with the minimum number of selected features d when the other criterions are equal.

$$f_3 = d \,. \tag{9}$$

Then, IFMOA, capable of maintaining a diverse population of solutions and resulting in global Pareto optimal solutions by multiobjective, is used to deal with the optimization problem with above 3 objectives.

4.2 A Review of IFMOA

IFMOA, an algorithm inspired by immune response, combines the Pareto-strength based fitness assignment strategy, the Clonal Selection Operator, and the Immune Forgetting Operator to solve multiobjective optimization problems [12]. Clonal Selection Operator (CSO) and Immune Forgetting Operator (IFO) are two operators simulating the dynamic process of immune response in IFMOA. They allow us to extend the searching scope and increase the diversity of the populations, resulting in more uniformly distributing global Pareto optimal solutions and more integrated Pareto fronts over the tradeoff surface.

IFMOA uses two populations: one of antibodies and another one of the nondominated solutions of each generation. The problems to be optimized are denoted as antigens, and the candidate solutions are denoted as antibodies. The affinity indicates the matching between the solution and the problem. The overall flow of IFMOA is described as follows:

```
Algorithm: The Immune Forgetting Multiobjective Optimization Algorithm
Step1: $k=0$, Generate the initial antibody population $A(0)=\{\mathbf{a}_1(0),\mathbf{a}_2(0),\cdots\mathbf{a}_n(0)\}$ and
       create an empty matrix (antibody archive) $\overline{\boldsymbol{P}}(0)$.
Step2: Assigning affinity to the individuals in the combinational population pool
       of antibody population $\boldsymbol{A}(k)$ and archive $\overline{\boldsymbol{P}}(k)$.
Step3: If the termination criterion is not satisfied, carry on the following operations, otherwise, stop.
       Step 3.1: Apply Clonal Selection Operator to antibody population $\boldsymbol{A}(k)$.
       Step 3.2: Update the antibody archive $\overline{\boldsymbol{P}}(k)$ and get $\overline{\boldsymbol{P}}(k+1)$.
       Step 3.3: Perform Immune Forgetting Operator and get $\boldsymbol{A}(k+1)$.
       Step 3.4: $k=k+1$ go to Step 2.
```

Fig. 2. Immune Forgetting Multiobjective Optimization Algorithm

4.3 Application of IFMOA in Unsupervised Feature Selection

Three problems must be solved first in the application of IFMOA to the unsupervised feature selection: encoding strategy, the construction of the affinity function to evaluate the performance of every individual, and the determinations of immune operators and the parameters.

4.3.1 Encoding

The decimal encoding is used and the length of an antibody is D+1, where D is the number of features. The initial antibody population A(0) is generated randomly at between 0 and 1. And each one of N_p (population size) antibodies comprises two parts, the feature saliency for each feature and the encoding for the number of clusters. Let $(a_{v_1}, a_{v_2}, \cdots, a_{v_D}, a_{v_{D+1}})$ denote an antibody, where $a_{v_i}, ..., a_{v_D}$ encode the feature saliency values of the associated features and $a_{v_{D+1}}$ denotes the number of clusters.

Features with large saliencies are kept to constitute a suitable feature subset. And optimizing feature saliency values naturally leads to feature selection. The number of clusters can be obtained by the following decoding method: $c = round(c_{\min} + (c_{\max} - c_{\min})a_{v_{D+1}})$ where $c_{\min}$ is the minimal number of clusters, $c_{\min} = 2$ since one cluster for a data set is meaningless, and $c_{\max}$ is the preestablished maximal number of clusters. $round(a)$ rounds the elements of a to the nearest integers.

In the evolutionary, the feature saliency value is likely to become small if one of the input features is unimportant for the clustering. Consequently, when the feature saliency value becomes small enough, it means that it is possible to remove the corresponding feature without sacrificing the performance of the clustering. In our work, all the feature saliency values $a_{v_i}, i = 1, 2, \cdots, D$ denoted by each individual are normalized first and let $\sum_{i=1}^{D} a_{v_i} = 1$. The threshold is set to be $1/D$ and the features whose feature saliency values are lower than the threshold are discarded.

4.3.2 Computation of Affinity

In IFMOA, each individual in the combinational population pool of antibody population $A(k)$ and archive $\overline{\boldsymbol{P}}(k)$ is assigned an affinity value $F(i)$, which is computed by $F(i) = R(i) + D(i)$. $R(i)$ is determined according to the concept of Pareto strength. $R(i) = 0$ indicates that the individual p_i is not dominated by any other individuals, corresponding to a nondominated solution. The purpose of $R(i)$ is to find the nondominated solutions. $D(i)$ is additionally incorporated to guide a more precise search process and is computed by $D(i) = 1/(d(i) + 1)$, where $d(i)$ is the sum of distances in the solution space between one antibody and its two nearest individuals. The smaller $D(i)$, the lower the similarity between one antibody and its neighboring individuals, which is beneficial to the diversity of the population. Thus the overall affinity should be minimized.

4.3.3 Immune Operations

Clonal Selection Operator(CSO)

Inspired by the clonal selection theory, the Clonal Selection Operator (CSO) implements clonal proliferation ($T_{\mathrm{P}}^{\mathrm{C}}$), affinity maturation ($T_{\mathrm{M}}^{\mathrm{A}}$) and clonal selection ($T_{\mathrm{S}}^{\mathrm{C}}$) on the antibody population $A(k)$. The evolvement process of CSO can be described as:

$$A(k) \xrightarrow{T_{\mathrm{P}}^{\mathrm{C}}} Y(k) \xrightarrow{T_{\mathrm{M}}^{\mathrm{A}}} (Z(k) \cup A(k)) \xrightarrow{T_{\mathrm{S}}^{\mathrm{C}}} A'(k) \tag{10}$$

According to CSO, an antibody $a_i = \{x_1, x_2, \cdots, x_m\}$ in the solution space will obtain q_i copies by performing clonal proliferation, and then we get the new population $Y(k)$. The affinity maturation is implemented through random changes, i.e.mutation of $Y(k)$, with mutation probability $P_m = 1/l$ where l is the length of an antibody. Such changes may lead to an increase in the affinity of the clonal antibody occasionally. But mutation is not applied to $A(k)$ in order to save the information of original antibody population. Thus the population after the affinity maturation is composed of the mutated population $Z(k)$ and the original population $A(k)$. Finally, clonal selection is used to update the antibody population. In this case, preference is given to the individual with the higher affinity between the original antibody and its q_i copies after mutation. CSO reproduces antibodies and selects their improved progenies, so each individual will be optimized locally and the newcomers yield a broader exploration of the search space.

Updating Antibody Archive
In the iterative of IFMOA, the antibody archive stores the nondominated solutions of current combinational population. Such external population is the elitist mechanism most commonly used in multiobjective optimization, and it allows us to move towards the true Pareto front of the multiobjective problem. We set the antibody archive not to exceed a fixed size N_f, and assume $\left|\overline{\boldsymbol{P}}(k+1)\right|$ is its actual size. If $\left|\overline{\boldsymbol{P}}(k+1)\right| \prec N_f$, copy $\left(N_f - \left|\overline{\boldsymbol{P}}(k+1)\right|\right)$ dominated individuals having the best affinity from the current combinational population to $\overline{\boldsymbol{P}}(k+1)$; or else, compress $\overline{\boldsymbol{P}}(k+1)$ until $\left|\overline{\boldsymbol{P}}(k+1)\right|$ is having the best affinity equal to N_f.

Updating Clonal Forgetting Unit
We randomly select r antibodies in the current antibody population to fill the clonal forgetting pool. The activation of clonal forgetting pool is implemented by replacing the whole clonal forgetting pool with individuals from the antibody archive. The value of r is decided by clonal forgetting ratio $T\%$, $r = round\left(T\% * n\right)$. $T\%$ is related to antibody population size and the problems to be optimized, it can be self-adaptive or fixed.

5 Experimental Results and Discussion

To evaluate the performance of the proposed method, a synthetic data set is used first, which consists of 400 data points and 6 features. With the first two significant features, the points form four well defined clusters as shown in the first figure of Fig.3, which are formed by generating points from a pseudo-Gaussian distribution. Features 3 and 4 are got by the similar way with feature 2. Features 5 and 6 are independently sampled from uniform distribution. Fig.3 illustrates this data set by projecting the points onto the feature subspaces with two dimensions.

In the experiment, the length of the individual is 9. The first 8 decimal numbers encode the features, while the remaining one for the number of clusters. In IFMOA, the population size is 20, the probability of mutation is 1/9, and the clonal forgetting proportion $T\%$ is set to be 0.08. Antibody archive size $N_f = 100$ and clonal scale $n_c = 5$. The termination criterion is triggered whenever the maximum number of generations 50 is attained. The maximal number of clusters is set to be 8.

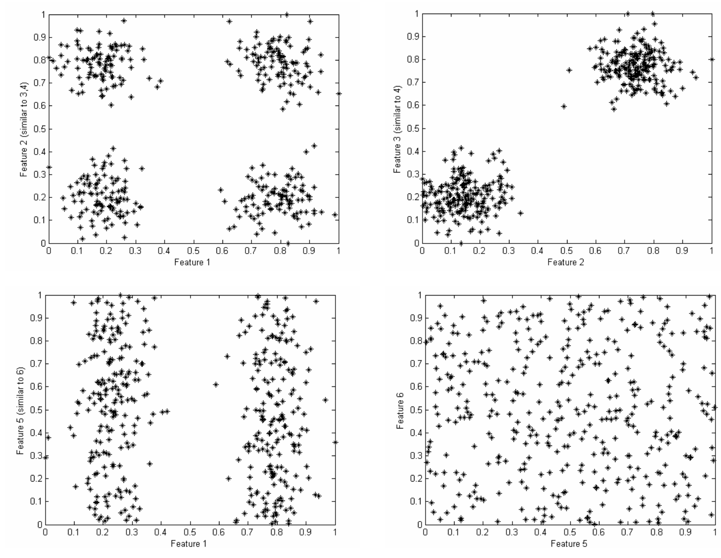


Fig. 3. Some 2-dimensional projections of the synthetic data set

Table 1 shows the set of nondominated solutions or Pareto solutions found by IFMOA. It can be observed that the solution found can describe the data set shown in Fig.3 well. Because the performance of clustering is our main focus here, then S_1 is the best solution in terms of the XB index.

Table 1. Nondominated solutions for the synthetic data

Solution	Feature subset	Num. of features	Num. of clusters	XB index
S1	$\{x_1, x_3\}$	2	4	0.1278
S2	$\{x_2, x_3, x_4\}$	3	2	0.1690
S3	$\{x_2, x_4, x_5\}$	3	2	0.1848

The second experiment is implemented to test the algorithm on the real data sets as shown in Table 2 from (http://www.ics.uci.edu/~mlearn/MLSummary.html) UCI machine learning repository. All the data sets are normalized beforehand.

In the experiment, each data set was first randomly divided into two halves: one for feature selection, another for testing. According to the selected features and the number of clusters on the former set by the proposed algorithm, each data point in the test set is assigning to the cluster that most likely generated it. We evaluate the results by interpreting the components as clusters and compare them with the ground truth labels. In each time, a final solution is selected among all the Pareto solutions in terms of XB index. In IFMOA, except the length of individual and the mutation are relevant

Table 2. Data sets used in experiment 2

Data set	Class	Attribute	Num. of data
wine	3	13	178
Ionospere	2	34	351
wdbc	2	30	569

to the dimension of data set, the values of the other parameters are same as those in experiment 1. The method is denoted as FSFCM-IFMOA in table 3.

For each data set, we perform the proposed method 10 runs and record the average error rate and the standard deviation on the test set. For comparison, we combine the three objectives in section 4.1 into one objective function by a random weighted summation, and the suitable feature subset and the number of clusters are achieved by minimizing the objective function using Immune Clonal Algorithm (ICA) [15]. The function is defined as $F = \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3$, in which α_i is randomly generated in each evaluation and $\sum_{i=1}^{3} \alpha_i = 1$. In ICA, the encoding strategy and the parameters including the population size, the probability of mutation, the clonal scale and the termination criterion are set the same as those in IFMOA. FSFCM-ICA is used for denoting the method and the average results of 10 runs are recorded in Table 3. Furthermore, we also ran FCM 10 times using all the features with the fixed number of clusters.

Table 3. Comparison of the error rates and the standard deviations of data sets with and without feature selection

Data set	Method	Error (%)	Number of clusters	Number of features
wine	FCM	5.51 ± 1.71	Fixed at 3	Fixed at 13
	FSFCM-ICA	7.87 ± 5.43	3.10 ± 0.57	6.4 ± 1.5
	FSFCM-IFMOA	5.05 ± 0.80	3.33 ± 0.48	9.0 ± 1.00
Ionospere	FCM	28.41 ± 2.99	Fixed at 2	Fixed at 34
	FSFCM-ICA	32.37 ± 1.50	2.40 ± 0.89	16.6 ± 1.52
	FSFCM-IFMOA	24.43 ± 3.46	2.33 ± 0.57	27.0 ± 1.00
wdbc	FCM	7.68 ± 1.24	Fixed at 2	Fixed at 30
	FSFCM-ICA	8.42 ± 1.75	5.00 ± 1.87	15.21 ± 0.84
	FSFCM-IFMOA	7.42 ± 1.58	2.30 ± 0.67	25.9 ± 1.29

From the results, we can find that the proposed method improved the performance in terms of the error rate when compared with using all the features and the fixed number of clusters. The improvement is more significant for the Ionospere data set. However, it is also shown that the dimension of the selected feature subset is high. It may be due to the fact that the final solution of each run is selected among all the Pareto solutions in terms of XB index since the performance is our preference. In addition, the value of the threshold for feature saliencies will influence the performance of feature selection also when the decimal encoding strategy is used and the number of selected features is unknown. It is also shown that the error rates of the FSFCM-ICA with randomly weighting are lower than the proposed method in our experiments.

6 Conclusion

In this paper, we proposed an unsupervised feature selection algorithm based on Immune Forgetting Multiobjective Optimization Algorithm. Unsupervised feature selection is considered as a combinational optimization problems for minimize the number of features used and the criterion for measuring the performance of clustering. At the same time, different feature subsets lead to varying cluster structures, namely, the number of clusters is relevant to the feature subset considered. Instead of combining these evaluations into one objective function, we make use of the multiobjective immune clonal algorithm with forgetting strategy to find the more discriminant features for clustering and the pertinent number of clusters. The results of experiments on synthetic data and real data sets show the potential of the method. Certainly, we just get some primary results and the application of the method on the segmentation of remote sensing image is our focus of further study.

Acknowledgements. The authors would like to thank the anonymous reviewers for valuable suggestions that helped to greatly improve this paper. This work was supported by the National Grand Fundamental Research 973 Program of China under Grant No.2001CB309403 and the Chinese National Natural Science Foundation under Grant No. 60372045, 60372050, 60133010 and 60472084.

References

1. Kohavi, R., John, G.H.: Wrappers for Feature Subset Selection. Artificial Intelligence Journal. 97 (1997) 273–324
2. Dash, M., Choi, K., Scheuermann,P., Liu, H.: Feature Selection for Clustering - A Filter Solution, In: Proceedings of 2nd IEEE International Conference on Data Mining (ICDM'02). (2002) 115–122
3. Mitra P., Murthy, C.A.: Unsupervised Feature Selection Using Feature Similarity. IEEE Trans. Pattern Analysis and Machine Intelligence. 24(2002): 301–312
4. Law, M.H.C., Figueiredo, Ma´ rio A.T., Jain, A.K.: Simultaneous Feature Selection and Clustering Using Mixture Models. IEEE Trans Pattern Analysis and Machine Intelligence. 26(2004): 1154–1166
5. Dy, J.G., Brodley, C.E., Kak, A., Broderick, L.S. and Aisen, A.M.: Unsupervised Feature Selection Applied to Content-Based Retrieval of Lung Images. IEEE Trans. Pattern Analysis and Machine Intelligence. 25(2003): 373–378
6. Morita, M., Sabourin, R., Bortolozzi, F., and Suen, C. Y.: Unsupervised Feature Selection Using Multi-Objective Genetic Algorithms for Handwritten Word Recognition. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition. (2003): 666–670

7. Kim, Y.S., Street, W.N. and Menczer, F.: Feature Selection in Unsupervised Learning via Evolutionary Search. In: Proc. 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2000) 365–369
8. Yang, J., Honavar, V.: Feature Subset Selection Using a Genetic Algorithm. IEEE Trans. on Intelligent Systems. 13 (1998) 44–49
9. Li, R., Bhanu, B., Dong, A.: Coevolutionary Feature Synthesized EM Algorithm for Image Retrieval. In: Proceedings of the Application of Computer Multimedia. (2005) 696–705
10. Lin, Y., Bhanu, B.: Evolutionary Feature Synthesis for Object Recognition. IEEE Trans. on Systems, Man, and Cybernetics–Part C. 35 (2005) 156–171
11. Zhang, X.R, Shan, T, Jiao, L.C.: SAR Image Classification Based on Immune Clonal Feature Selection. In: Aurélio, C.C, Mohamed, S.K. (Eds.): Proceedings of Image Analysis and Recognition. Lecture Notes in Computer Science, Vol. 3212. Springer-Verlag, Berlin Heidelberg New York (2004) 504–511
12. Lu, B., Jiao, L.C., Du, H.F., Gong, M.G.: IFMOA: Immune Forgetting Multiobjective Optimization Algorithm. In: Lipo Wang, Ke Chen, and YewS. Ong(Eds.): Proceedings of the First International Conf. on Advances in Natural Computation. Lecture Notes in Computer Science, Vol. 3612. Springer-Verlag, Berlin Heidelberg New York (2005) 399–408
13. Xie, X.L., Beni, G.: A Validity Measure for Fuzzy Clustering. IEEE Trans. Pattern Analysis and Machine Intelligence. 13 (1991) 841–847
14. Fraley, C., Raftery, A.E.: How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. The Computer Journal. 41(1998) 578–588
15. Du, H.F., Jiao, L.C., Wang S.A.: Clonal Operator and Antibody Clone Algorithms. In: Proceedings of the First International Conference on Machine Learning and Cybernetics. Beijing. (2002) 506–510

Classifying and Counting Vehicles in Traffic Control Applications

Francesco Archetti, Enza Messina, Daniele Toscani, and Leonardo Vanneschi

Dipartimento di Informatica, Sistemistica e Comunicazione (D.I.S.Co.),
University of Milano-Bicocca,
Milan, Italy
{archetti, messina, toscani, vanneschi}@disco.unimib.it

Abstract. This paper presents a machine learning system to handle traffic control applications. The input of the system is a set of image sequences coming from a fixed camera. The system can be divided into two main subsystems: the first one, based on Artificial Neural Networks classifies the typology of vehicles moving within a limited image area for each frame of the sequence; the second one, based on Genetic Algorithms, takes as input the frame-by-frame classifications and reconstructs the global traffic scenario by counting the number of vehicles of each typology. This task is particularly hard when the frame rate is low. The results obtained by our system are reliable even for very low frame rate (i.e. four frames per second). Our system is currently used by a company for real-time traffic control.

1 Introduction

The problem of counting and classifying in an automatic, reliable and efficient way the vehicles that travel along an urban street or a motorway is becoming more and more important to maintain traffic safety and fluency (see for instance [7, 1, 8, 3, 13]). Companies are beginning to employ systems based on filmed sequences, since they appear to be more flexible, more maintainable and less expensive than systems based on technological supports, such as sensors or other analogous devices that have to be physically installed in many different places on the street. While, in order to contain costs, some companies use less sophisticated devices, which produce image sequences at a lower frame rate, analyzing low frame rate sequences by means of computer systems can be very difficult. Several commercial systems for traffic surveillance based on filmed sequences exist (see among the others CVCS, by Alvarado Manufacturing [9], CCATS by Traficon and AUTOSCOPE by Econolite). However these systems seem to be based on high frame rate image sequences. In [2], Eikvil and Huseby pointed out the main limitations of these systems and proposed a new system based on Hidden Markov Models (HMM). As pointed out in [12, 13], Eikvil's and Huseby's system has excellent performances when applied to high frame rate image sequences, but it fails to correctly reconstruct the traffic scenario in the presence of low frame rate ones. In [12] a good discussion on the motivations for which a system based on HMM has poor performances on low frame rate image sequences can be found. In this paper, we present a system for traffic control which is able to work in real-time on low frame rate image sequences. This work is a part of a joined research project between the University of Milano-Bicocca and the Comerson company (http://www.comerson.com). The main goal is recognizing and counting different types of vehicles that have traveled along a street in a given time frame. In order to calculate statistics about the size of vehicles traveling and thus propose strategies to increase traffic safety and efficiency, we distinguish three types of vehicles: *cars*, *trucks* and *motorbikes*. Our system is composed by two modules. The first one, called *Vehicle Classifier System*, is based on a standard Feed Forward Neural Network (NN) trained with the Backpropagation learning rule [5, 11]. The second one, called *Vehicle Counting System*, is based on Genetic Algorithms (GAs) [6, 4]. This paper is organized as follows: in section 2, the Vehicle Classifier System is presented and some experimental results are discussed. Section 3 describes the Vehicle Counting System Experimental results are presented also for it. Finally, section 4 offers our conclusions and hints for future work.

2 Vehicle Classifier System

This system takes as input a sequence of images (or frames). These images are interpreted as matrix of pixels, where every pixel can take values over the range $[0, 255]$, corresponding to the various tonalities of gray (0 = white, 255 = black). The analysis of a sequence of consecutive images (or frames) allows us to extract the background with the same technique as the one used in [2]. Subsequently, a number of *virtual sensors* (or *detectors*) of rectangular shape, arranged in a grid structure, are placed in some limited regions of the street. For every detector S_i composing the grid ($1 \leq i \leq 36$ in figure) let $s_{(i,k)}$ be the sum of the squared pixel-by-pixel differences between frame k (the running image) and the background. The training set that is supplied as input to the classifier can then be represented as a set of sequences $s_{(1,j)}, s_{(2,j)} \ldots, s_{(36,j)}, C_j$, with $1 \leq j \leq N$, where N is the number of frames in the image sequence. and, for each j, the value of C_j, is interpreted as the class corresponding to the pattern $s_{(1,j)}, s_{(2,j)}, \ldots, s_{(36,j)}$. This class can be: c, t, M, e or m if a car, a truck, a motorbike, no vehicle or more than one vehicle is present on the grid of detectors, respectively. This system demands a training phase in which a human being supplies the vehicle class contained into the grid of detectors at each instant. Here, we present some experiments on the three different street scenarios (whose names are *Noranco*, *Bollate* and *Saragozza*). Results obtained by *k-folds cross-validation* are reported in table 1 (see for instance [10] for a definition of the *cci*, *p* and *r* measures and for a description of the *k-folds cross-validation* method).

3 Vehicle Counting System

The output produced by the Vehicle Classifier System is a sequence like $G_1, G_2 \ldots, G_n$, where $\forall i \in [1, n], G_i \in \{C, T, M, m, e\}$. A sequence of identical symbols, like for instance (C, C, C, C, C, C). may represent: (1) the presence of *the same* car on the grid for six consecutive frames, or (2) the presence of a number of *different* cars in the different frames. The second of these events can happen because of the low frame frequency of

Table 1. Results of the Vehicle Classifier System on three different image sequences. Each row of this table corresponds to a different sequence. Names of the sequences are reported in column 1. Column 2 shows results of the cumulative correctly classified instances (*cci*) for all the vehicles. Columns 3 to 12 report results of precision (p) and recall (r) for the classes Car, Truck, Motorbike, Empty and Mix respectively.

Image Sequence	*cci*	Car		Truck		Motorbike		Empty		Mix	
		p	r	p	r	p	r	p	r	p	r
Noranco	96.724%	0.837	0.802	0.899	0.873	0	0	0.984	0.991	0	0
Bollate	95.283%	0.904	0.906	0.773	0.791	0	0	0.976	0.988	0.647	0.324
Saragozza	90.42%	0.909	0.923	0.666	0.563	0.754	0.325	0.93	0.952	0.222	1.0

our image sequences. The task of counting the vehicles strongly depends on the identification of which one of these events happened. The Vehicle Counting System can be partitioned into three phases: preprocessing, GA and statistics (and model) generation. These phases are described below.

Preprocessing Phase. Let a sequence returned by the Vehicle Classifier System be $S = S_0, S_1, ..., S_{n-1}$. In this phase, it is transformed into a sequence: $V = V_0, V_1, ..., V_{k-1}$, where each symbol in the V sequence "points" at one or more consecutive identical symbols in the S sequence. All the m and e symbols in the S sequence are not reported in the V sequence, but their information is not lost: for instance, if the V sequence contains two consecutive symbols CC, then we are sure that they don't correspond to the same car: the presence of m and e symbols in the S sequence allows us to build a V sequence where consecutive identical symbols surely do not correspond to the same vehicle.

Genetic Algorithm. Let $V = [V_0, V_1, ..., V_{k-1}]$ be the output of the preprocessing phase. The GA associates V with a population of individuals of the form $I = [N_0, N_1, ..., N_{k-1}]$, such that: (1) $\forall i \in [0, k-1]$, N_i is the number of different vehicles of type V_i that have passed in the i^{th} sequence. (2) $\forall i \in [0, k-1]$, $1 \leq N_i \leq L_i$, where L_i is the length of the substring of the S sequence "pointed" by V_i. The fitness of GA individuals is calculated by supervisioning a portion of the image sequence, i.e. by manually counting how many cars, trucks and motorbikes pass through the grid of detectors. Let X_j be the number of cars, Y_j be the number of trucks and Z_j be the number of motorbikes that really passed through the grid (obtained by means of this supervision) during a given subsequence j. The fitness of an individual $I = [N_0, N_1, ..., N_{k-1}]$ is: $f(I) = \sum_{j=1}^{h} (\frac{1}{X_j} |X_j - \sum_{i:V[i]=C} N_i| + \frac{1}{Y_j} |Y_j - \sum_{i:V[i]=T} N_i| + \frac{1}{Z_j} |Z_j - \sum_{i:V[i]=M} N_i|)$, where h is the total number of subsequences the image sequence has been partitioned into. Normalizations of each member of the sum have been done in order to give the same importance to each vehicle independently from the number of instances of that vehicle which have been observed. Standard crossover operator can be used, while the mutation works by replacing the old value of N_i by a uniformly generated random number included in $[1, L_i]$.

Statistics and model generation. Let S be the sequence produced by the Vehicle Classifier System, V the one produced by the preprocessing phase and W the one returned by the GA. The model of our system is generated from the output of the GA, by considering the rates of all the results assigned to each subsequence of the same number of identical symbols. For instance, let V contain a C symbol in positions 2, 5 and 11. Let each one these three C symbols in V point to a sequence of 4 consecutive C symbols in S. Furthermore, let no other C symbol in V point to a sequence of 4 consecutive C symbols in S. Finally W contain the numbers 2 in positions 2 and 11 and 1 in position 5. Then, a consecutive sequence of 4 C symbols will be considered by our system as 2 cars with probability $2/3$ and as 1 car with probability $1/3$.

Experimental Results. The performance of the Vehicle Counting System has been tested on the same street scenarios as the ones described in section 2. The learning phase has always been done on a three minutes long image sequence of the same scenario as the one used during the tests. In that phase, the GA runs have been executed with the following parameters: population size: 100, crossover rate: 95%, mutation rate: 0.001%, tournament selection of size 10, chromosome length depending on the size of the V sequence produced by the preprocessing phase. Results are reported in table 2, where

Table 2. Results of the Vehicle Counting System for the same image sequences as in section 2. Sequences have been partitioned into two subsequences. Both subsequences and the whole sequences have been tested. Each line is related to one of these subsequences. Column 2 reports the set of frames corresponding to each subsequence. Columns 3 to 8 report the real number and the counted number of vehicles for the three typologies considered here (cars, trucks and motorbikes).

Image Sequence	Frame set	Car		Truck		Motorbike	
		real	counted	real	counted	real	counted
Noranco	0-2730	67	73	25	20	1	0
Noranco	2730-4958	60	62	17	16	0	0
Noranco	0-4958	127	133	42	34	1	0
Bollate	0-4050	159	153	2	6	3	5
Bollate	4050-7579	138	142	4	2	2	1
Bollate	0-7579	287	293	6	9	5	8
Saragozza	0-2615	84	89	0	0	5	3
Saragozza	2615-5530	77	82	2	1	4	1
Saragozza	0-5530	161	173	2	1	9	6

the true values of the numbers of cars, trucks and motorbikes passed through the street (values counted "by hand") are shown next to the values counted by the Vehicle Counting System. Each image sequence has been divided into two subsequences of frames. The frame numbers corresponding to each subsequence is shown in column two. For each image sequence, experiments have been performed on the whole sequence and on the two subsequences. As the table shows, the numbers of cars, trucks and motorbikes counted by the Vehicle Counting System very well approximate the true values for all the cases considered.

4 Conclusions and Future Work

A new system for real-time classifying and counting vehicles has been presented in this paper. It is based on the analysis of low frame rate image sequences of a street scenario over a given time frame, taken by a fixed camera. It is composed by two subsystems: one, based on Feed Forward Neural Networks, for classifying the typology of the moving vehicles. The other, based on Genetic Algorithms (GAs), for counting them. Results shown in this paper are encouraging. Future work includes studying how the preprocessing performances can affect the reliability of the whole system.

Acknowledgments. The authors gratefully acknowledge the Comerson company for providing data for the experiments shown in this paper.

References

1. N. Papanikolopoulos E. Wahlstrom, O. Masoud. Monitoring driver activities, Report no. CTS 04-05. Intelligent Transportation Systems Institute. Internal Report. 2004. http://www.its.umn.edu/research/complete.html.
2. L. Eikvil and R. B. Huseby. Traffic surveillance in real-time using hidden markov models, Scandinavian Conference on Image Analysis, SCIA01. 2001. http://citeseer.ist.psu.edu/eikvil01traffic.html.
3. G. Giuliano, J. Heidemann, M. Hu, F. Silva, and X. Wang. Rapidly deployable sensors for vehicle counting and classification, 2004. Internal report of the I-LENSE: ISI Laboratory for Embedded Networked Sensor Experimentation.
4. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
5. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, London, UK, 1999.
6. J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975.
7. J. Hourdakis, T. Morris, P. Michalopoulos, and K. Wood. Advanced portable wireless measurement and observation station, Report no. CTS 05-07. Intelligent Transportation Systems Institute. Internal Report. 2005. http://www.its.umn.edu/research/complete.html.
8. V. Kwigizile, M. F. Selekwa, and R. Mussa. Highway vehicle classification by probabilistic neural networks. In V. Barr and Z. Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*. AAAI Press, 2004.
9. ALVARADO Manufacturing. Computerized vehicle counting system, 2005. Internal report.
10. T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1996.
11. E. Oja. Principal component analysis. In Michael A. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 753–756. MIT Press, 1995.
12. Consorzio Milano Ricerche. Reti neurali per la classificazione di immagini e il riconoscimento di veicoli, 2005. Status Report Progetto Comerson. Internal Report.
13. L. Vanneschi and F. Archetti. Resolving temporal sequences for traffic control applications with genetic algorithms. In *Proceedings of the 7th International Conference on Artificial Evolution (EA '05)*, 2005.

A Neural Evolutionary Classification Method for Brain-Wave Analysis

Antonia Azzini and Andrea G.B. Tettamanzi

Università degli Studi di Milano,
Dipartimento di Tecnologie dell'Informazione,
via Bramante 65, I-26013, Crema, Italy
{azzini, tettamanzi}@dti.unimi.it

Abstract. This paper presents an approach to the joint optimization of neural network structure and weights which can take advantage of backpropagation as a specialized decoder. The approach is applied to binary classification of brain waves in the context of brain-computer interfaces.

Keywords: Neural Networks, Classification, Brain-Computer Interfaces, Evolutionary Algorithms.

1 Introduction

The evolutionary approach that implements the conjunction of evolutionary algorithms (EAs) with neural networks (NNs) is a more integrated way of designing ANNs since it allows all aspects of NN design to be taken into account at once and does not require expert knowledge of the problem. Some EAs have implemented a search over the topology space, or a search for the optimal learning parameters or weight setting.

The primary motivation for using evolutionary techniques to establish the weighting values rather than traditional gradient descent techniques such as backpropagation (BP) [5], lies in the trapping in local minima and in the non-differentiability of the function. For this reason, rather than adapting weights based on local improvement only, EAs evolve weights based on the whole network fitness. An interesting area of evolutionary NNs is the combination of architecture and weight evolution in order to find an optimal network architecture and to train the network on a given data set. The advantage of combining these two basic elements of a NN is that a completely functioning network can be evolved without any intervention by an expert.

2 The Neuro-genetic Approach

The approach is designed to be able to take advantage of the backpropagation (BP) algorithm if that is possible and beneficial; however, it can also do without it. This research was tested by an industrial application [1] for the design of neural engine controllers, with particular attention to reduced power consumption and silicon area. In this work we apply our neuro-genetic approach to brain wave signal processing, in particular as a classification algorithm in the analysis of P300 Evoked Potential. We restrict our attention to Multi-Layer Perceptrons (MLPs), a specific subset of the feedforwards NNs which is powerful enough to be general, while at the same time allowing for a compact representation.

2.1 The Evolutionary Representation

The initial population is seeded with random networks initialized with different hidden layer sizes, using two exponential distributions to determine the number of hidden layers and neurons for each individual, and a normal distribution to determine the weights and bias values. For all weights matrices and bias we also define matrices of variance, that will be applied in conjunction with evolutionary strategies in order to perturb network weights and bias. Variance matrices will be initialized with matrices of all ones. In both cases, unlike other approaches like [6], the maximum size and number of the hidden layers is neither pre-determined, nor bounded, even though the fitness function may penalize large networks. Each individual is encoded in a structure in which we maintain basic information about string codification of topology and weights and bias matrices. This kind of structure is defined together with all parameters algorithms in [1]. The values of all these parameters are affected by the genetic operators during evolution, in order to perform incremental (adding hidden neurons or hidden layers) and decremental (pruning hidden neurons or hidden layers) learning.

2.2 Fitness

Like indicated in [1] the fitness is proportional to the value of the mse and to the cost of the considered network. It is defined as

$$f = \lambda kc + (1 - \lambda)\text{mse}, \tag{1}$$

where $\lambda \in [0, 1]$ is a parameter which specifies the desired trade-off between network cost and accuracy, k is a constant for scaling the cost and the mse of the network to a comparable scale, and c is the overall cost of the considered network, defined as

$$c = \alpha N_{hn} + \beta N_{syn}, \tag{2}$$

where N_{hn} is the number of hidden neurons, and N_{syn} is the number of synapses. The mse depends on the *Activation Function*, that calculates all the output values for each single layer of the neural network. In this work we use the *Sigmoid Transfer Function*. To be more precise, two fitness values are actually calculated for each individual: the fitness f, used by the selection operator, and a test fitness $\hat{f}$. $\hat{f}$ is calculated according to Equation 1 by using the mse over the test set. When BP is used, i.e., if $bp = 1$, $f = \hat{f}$; otherwise ($bp = 0$), f is calculated according to Equation 1 by using the mse over the training and test sets together.

2.3 Genetic Operators

The genetic core of the algorithm is described by the following pseudo-code:

```
1. Select from the population (of size $n$) $\lfloor n/2 \rfloor$ individuals by truncation and
   create a new population of size $n$ with copies of the selected individuals.
2. For all individuals in the population:
   (a) Mutate the weights and the topology of the offspring.
   (b) Train the resulting network using the training and test sets if $bp = 1$.
   (c) Calculate $f$ and $\hat{f}$.
   (d) Save the individual with lowest $\hat{f}$ as the best-so-far individual if the $\hat{f}$
       of the previously saved best-so-far individual is higher (worse).
3. Save statistics.
```

The *selection* strategy used by the algorithm is truncation: starting from a population of n individuals, the worst $\lfloor n/2 \rfloor$ (with respect to f) are eliminated. The remaining individuals are duplicated in order to replace those eliminated. Finally, the population is randomly permuted. Two types of *mutation* operators are used: a general random perturbation of weights, applied before the BP learning rule, and three *mutation* operators which affect the network architecture. The weight mutation is applied first, followed by the topology mutations, as follows:

1. Weight mutation: all the weight matrices and the biases are perturbed by using equations based on variance matrices and evolutionary strategies applied to the number of synapses of the entire neural network. These equations are described in detail in [1]. After this perturbation has been applied, neurons whose contribution to the network output is negligible are eliminated: a variable threshold is defined, depending on a norm (in our case L_∞) of the weight vector for each node, and the relevant average and standard deviation of the norms of the considered layer.
2. Topology mutations: these operators affect the network structure (i.e., the number of neurons in each layer and the number of hidden layers). In particular, three mutations can occur, which are described in detail in [1]:
   (a) Insertion of one hidden layer with probability p^+_{layer};
   (b) Deletion of one hidden layer with probability p^-_{layer};
   (c) Insertion of a neuron with probability p^-_{layer}.

All three topology mutation operators are designed so as to minimize their impact on the behavior of the network; in other words, they are designed to be as little disruptive (and as much neutral) as possible.

3 Application to Brain-Wave Analysis

3.1 Brain-Computer Interfaces and Problem Description

Brain Computer Interfaces (BCI) represent a new communication option for those suffering from neuromuscular impairment that prevents them from using conventional input devices, such as mouses, keyboards, joysticks, etc. Exploiting the residual functions of the brain, BCI may allow those patients to communicate. One of the most utilized electrical activities of the brain for BCI is the P300 Evoked Potential wave. This wave is a late-appearing component of an Event Related Potential (ERP) which can be auditory, visual or somatosensory. The idea of Donchin's solution [3] is that the patient is able to generate this signal without any training. This is due to the fact that the P300 is the brain's response to an unexpected or surprising event and is generated naturally. Donchin developed a BCI system able to detect an elicited P300 by signal averaging techniques (to reduce the noise) and used a specific method to speed up the overall performance. Donchin's idea has been adopted and further developed by Beverina and colleagues of ST Microelectronics [2]. We have applied the neuro-genetic approach described in Section 2 to the same dataset of P300 evoked potential used by Beverina and colleagues for their approach on brain signal analysis based on support vector machines.

3.2 Experimental Protocol and Results

The dataset provided by Beverina and colleagues consists of 700 negative cases and 295 positive cases. The feature are based on wavelets, morphological criteria and power in different time windows, for a total of 78 real-valued input attributes and 1 binary output attribute, indicating the class (positive or negative) of the relevant case. In order to create a balanced dataset of the same cardinality as the one used by Beverina and colleagues, for each run of the evolutionary algorithm we extract 218 positive cases from the 295 positive cases of the original set, and 218 negative cases from the 700 negative cases of the original set, to create a 436 case training dataset; for each run, we also create a 40 case test set by randomly extracting 20 positive cases and 20 negative cases from the remainder of the original dataset, so that there is no overlap between the training and the test sets. This is the same protocol followed by Beverina and colleagues. For each run of the evolutionary algorithm we allow up to 25,000 network evaluations (i.e., simulations of the network on the whole training set), including those performed by the backpropagation algorithm. 100 runs of the neuro-genetic approach with parameters set to their defaults values were executed with $bp = 0$ and $bp = 1$, i.e., both without and with backpropagation. The results obtained are presented in Table 1.

Table 1. Error rates of the best solutions found by the neuro-genetic approach with and without the use of backpropagation, averaged over 100 runs.

bp	training				test			
	false positives		false negatives		false positives		false negatives	
	avg	stdev	avg	stdev	avg	stdev	avg	stdev
0	93.28	38.668	86.14	38.289	7.62	3.9817	7.39	3.9026
1	29.42	14.329	36.47	12.716	1.96	1.4697	2.07	1.4924

Due to the way the training set and the test set are used, it is not surprising that error rates on the test sets look better than error rates on the training sets. That happens because, in the case of $bp = 1$, the performance of a network on the test set is used to calculate its fitness, which is used by the evolutionary algorithm to perform selection. Therefore, it is only networks whose performance on the test set is better than average which are selected for reproduction. The best solution has been found by the algorithm using backpropagation and is a multi-layer perceptron with one hidden layer with 4 neurons, which gives 22 false positives and 29 false negatives on the training set, while it commits no classification error on the test set. The results obtained by the neuro-genetic approach, without any specific tuning of the parameters, appear promising. To provide a reference, the average number of false positives obtained by Beverina and colleagues with support vector machines are 9.62 on the training set and 3.26 on the test set, whereas the number of false negatives are 21.34 on the training set and 4.45 on the test set [4].

4 Conclusion and Future Works

We illustrated an evolutionary approach to the joint design of neural network structure and weights which can take advantage of BP as a specialized decoder. The approach has been applied to the analysis of brain waves and compared to a mature approach based on support vector machines which has been presented in [2]. The comparison shows that our approach has some potential, even though, unsurprisingly, it does not attain the same levels of accuracy. Further work on this problem will include an in-depth study for parameters tuning and data set up, in order to improve the accuracy of classification.

References

1. A. Azzini, M. Lazzaroni, and G.B. Tettamanzi. A neuro-genetic approach to neural network design. In F. Sartori, S. Manzoni, and M. Palmonari, editors, *AI*IA 2005 Workshop on Evolutionary Computation*. AI*IA, Italian Association for Artificial Intelligence, September 20 2005.
2. F. Beverina, G. Palmas, S.Silvoni, F. Piccione, and S. Giove. User adaptive bcis: Ssvep and p300 based interfaces. *PsychNology Journal*, 1(4):331–354, 2003.
3. E. Donchin, K.M. Spencer, and R. Wijesinghe. The mental prosthesis: assessing the speed of a p300-based brain-computer interface. *IEEE Transactions on Rehabilitation Engineering*, 8(2):174–179, June 2000.
4. Giorgio Palmas. Personal communication, November 2005.
5. D. E. Rumelhart, J. L. McClelland, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
6. X. Yao and Y.Liu. Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, 91(1):83–90, 1998.

Differential Evolution Applied to a Multimodal Information Theoretic Optimization Problem

Patricia Besson, Jean-Marc Vesin, Vlad Popovici, and Murat Kunt

Signal Processing Institute, EPFL, Lausanne 1015, Switzerland
{patricia.besson, jean-marc.vesin, vlad.popovici, murat.kunt}@epfl.ch

Abstract. This paper discusses the problems raised by the optimization of a mutual information-based objective function, in the context of a multimodal speaker detection. As no approximation is used, this function is highly nonlinear and plagued by numerous local minima. Three different optimization methods are compared. The Differential Evolution algorithm is deemed to be the best for the problem at hand and, consequently, is used to perform the speaker detection.

1 Introduction

This paper addresses the optimization of an information theoretic-based function for the detection of the current speaker in audio-video (AV) sequences. A single camera and microphone are used so that the detection relies on the evaluation of the AV synchronism. As in [1], the information contained in each modality is fused at the feature level, optimizing the audio features with respect to the video ones. The objective function is based on mutual information (MI) and is highly nonlinear, with no analytical formulation of its gradient, unless approximations are introduced. The local Powell's method [2], has been tried in a first set of experiments and the conclusion was that a global approach was more suited. For this, two Evolutionary Algorithms (EAs) - the Genetic Algorithm in Continuous Space [3] and the Differential Evolution [4] - have been applied and their performance compared and analyzed.

After a brief introduction to the optimization problem, the three previously mentioned optimization methods are applied to the problem at hand. A detailed discussion follows the experiments presented in the last part of the paper and suggests that DE is the best choice for solving the given problem.

2 Multimodal Speaker Detection Framework

Theoretic framework for multimodal feature extraction. The detection of the current speaker in an AV sequence can be seen as a multimodal classification problem [1]. The goal is to estimate the class membership O ("speaker" or "non-speaker") of the bimodal source S that emits jointly an audio and a video signal, A and V. The probability P_e of assigning S to the wrong class should be minimized. Since a single camera and microphone are used, the detection rely on the evaluation of the synchronism between the AV signals. The estimation of the MI between the AV features can be used as such a classifer. This MI classifier must be provided with suitable features to perform well. Let F_A and F_V be the features extracted from A and V, and let $e = I(F_A,F_V)/H(F_A,F_V) \in [0,1]$ be the features' efficiency coefficient [1] (I and H standing respectively for Shannon's MI and entropy). Then the following inequality can be stated [1]:

$$P_e \geqslant 1 - \frac{e(F_A, F_V) + 1}{\log 2}. \tag{1}$$

By minimizing the right hand term of inequality (1), we expect to recover in each modality the information that originates from the common source S while discarding the source-independent information present in each signal. This would lead to a more accurate classification. For more details, see [1].

Application to speaker detection. The video features are the magnitude of the optical flow estimated over T frames in the mouth regions (rectangular regions including the lips and the chin). $T-1$ video feature vectors $F_{V,t}$ ($t = 1,\ldots,T-1$) are obtained, each element of these vectors being an observation of the random variable (r.v.) F_V.

The speech signal is represented as a set of $T-1$ vectors $\boldsymbol{C_t}$, each containing P Mel-Frequency Cepstral Coefficients (MFCCs), discarding the first coefficient.

Audio feature optimization. As mentioned, the goal is to construct better features for classification. The focus is now on the audio features $F_{A,t}(\boldsymbol{\alpha})$, associated to the r.v. F_A, that are built as the linear combination of the P MFCCs, $F_{A,t}(\boldsymbol{\alpha}) = \sum_{i=1}^{P} \boldsymbol{\alpha}(i) \cdot C_t(i)$, $\forall t = 1,\ldots,T-1$. The $\boldsymbol{\alpha}$ are to be optimized with respect to the Efficiency Coefficient Criterion (*ECC*):

$$\boldsymbol{\alpha}_{opt} = \arg\max_{\boldsymbol{\alpha}} \{e(F_V, F_A(\boldsymbol{\alpha}))\}. \tag{2}$$

A $\varDelta ECC$ criterion is introduced to perform only one optimization for two mouths and to take into account the discrepancy between the marginal densities of the video features in each region. If $F_V^{M_1}$ and $F_V^{M_2}$ denote the r.v. associated to regions M_1 and M_2 respectively, then the optimization problem becomes:

$$\boldsymbol{\alpha}_{opt} = \arg\max_{\boldsymbol{\alpha}} \left\{ [e(F_V^{M_1}, F_A(\boldsymbol{\alpha})) - e(F_V^{M_2}, F_A(\boldsymbol{\alpha}))]^2 \right\}. \tag{3}$$

This MI-based optimization criterion requires the availability of the probability density function (pdf) as well as of the marginal distributions of F_A and F_V. To avoid any restrictive assumption, they are estimated using Parzen windowing.

3 Optimization Method

Definition of the optimization problem. The extraction of the optimized audio features requires to find the vector $\boldsymbol{\alpha} \in \mathbb{R}^P$, that minimizes -$\varDelta ECC$ (Eq. (3)). To restrain the set of possible solutions, additional constraints have been introduced on the $\boldsymbol{\alpha}$ weights:

$$0 \leq \boldsymbol{\alpha}(i) \leq 1 \qquad \forall i = 1, 2, \ldots, P\ , \tag{4}$$

$$\sum_{i=1}^{P} \boldsymbol{\alpha}(i) = 1. \tag{5}$$

The optimization problem is highly nonlinear and gradient-free. Indeed, the MI-based objective function is *a priori* non-convex and is very likely to present rugged surface. Moreover, it is difficult to obtain an analytical form of its gradient due to the unknown form of the pdf of the extracted audio features. The use of Parzen window to estimate the pdf reduces the risk of getting trapped in a local minimum by smoothing the cost function. Because a trade-off has to be found between smoothness and accuracy of the distribution estimates, the smoothing parameter is iteratively adapted. Thus, the optimization problem is solved using a multi-resolution scheme (see [5]).

The deterministic Powell's method [2] has been used in a first set of experiments [6]. However, the objective function still exhibited too many local optima for a local optimization method to perform well. A global optimization strategy fulfilling the following requirements turned out to be preferable:

1. Efficiency for highly nonlinear problems without requiring the objective function to be differentiable or even continuous over the search space;
2. Efficiency with cost functions that present a shallow, rough error surface;
3. Ability to deal with real-valued parameters;
4. Efficiency in handling the two constraints defined by Eqs. (4, 5);

Genetic Algorithm in Continuous Space (GACS). An evolutionary approach such as GACS answers the first three requirements while presenting flexibility and simplicity of use in a challenging context. The adaptation developed in [3] efficiently deals with finite solution domain by relating the genetic operators to the constraints on the solution parameters. The crossover operator is defined such that the child chromosome is guaranted to lie into the acceptance domain (defined by Eq. (4)) provided its parents are valid. The mutation is performed by perturbing a randomly selected chromosome element with a zero-mean Gaussian perturbation which variance σ is defined as a certain fraction of the acceptance domain. The mutation is rejected if the mutated gene lies outside its acceptance domain. To satisfy the constraint defined by Eq. (5), the new population is normalized. Notice that the initial chromosomes are regularly placed in the acceptance domain according to a user-defined number of quantization levels Q [7]. This ensures a better initial exploration of the search space than a random initialization.

This extension of GACS leads to better results than the Powell's method. However, the mutation operator appears to be ineffective. The solutions are indeed very close to the search space limits. A high number of mutations are then rejected, resulting in a loss of the population diversity and in a premature convergence of the algorithm. The perturbation should adapt to the population evolution and should lead to a better exploration of the search space.

Differential Evolution (DE). Differential Evolution, introduced in [4], is an Evolution Strategy where the perturbation corresponds to the difference of chromosomes (or *vectors* in this context) randomly selected from the population. In this way, the distribution of the perturbation is determined by the distribution of the vectors themselves and no *a priori* defined distribution is required. Since this distribution depends primarily on the response of the population vectors to the objective function topography, the biases introduced by DE in the random walk towards the solution match those implicit in the function it is optimizing [8]. The exact algorithm we used is based on the so-called

DE/rand/1/bin algorithm [8]. The initial population however is generated as done with GACS. The validity of each perturbed vector is verified before starting the decision process. If the element j of a child vector i does not belong to the acceptance domain, it is replaced by the mean between its pre-mutation value and the bound that is violated [8]. This scheme is more efficient than the simple rejection adopted with GACS. Indeed, it allows to asymptotically approach the search space bounds. To handle the second constraint (Eq. (5)), a simple normalization is performed on each child vector, as it was done with GACS.

4 Results

Comparison of the optimization methods. All the test sequences are 4 seconds long, PAL standard ($T = 100$); 12 MFCCs are computed using 23.22ms Hamming windows. The three different optimization methods are tested on a single speaker sequence using -ECC (Eq. (2)) as the objective function. σ is fixed to 10% of the acceptance domain for GACS, while the scaling factor F and the crossover probability CR required by the DE algorithm [8] are fixed to 0.5 and 1 respectively [1]. Both algorithms are run for 400 generations on a population of 125 vectors. 33 runs were then performed with GACS and DE methods, whereas different initial solution guesses were tried for Powell's method. Table 1 summarizes the results. Obviously, a much better result is obtained using the

Table 1. Values of the *-ECC* cost function for 33 runs under the same conditions, on the same AV sequence

	Best Value	Mean Value	Standard Deviation
Powell	-0.0213	-0.0183	0.0047
GACS	-0.0695	-0.0619	0.0052
DE	-0.0788	-0.0773	0.0018

global optimization schemes instead of the local one. DE is the algorithm that reaches the best solution in a more stable way. Indeed, the standard deviation of the solutions is much smaller in the case of DE than in the case of the other two methods, giving us more confidence in the results. While the high variation of the solutions found with Powell's method is not a surprise (as it is very sensitive to initial conditions), the instability of GACS solution seems intriguing. However, this is less surprising when we analyze the evolution of the algorithm towards the solution: the degeneration of the population combined with the less systematic exploration of the solution space (especially the boundaries) make GACS solutions to be very different from run to run. Both the generation of the perturbation increment using the population itself instead of a predefined probability density and the handling of the out-of-range values allow the DE algorithm to achieve outstanding performance in the context of our problem.

Audiovisual speaker detection results. Five home-grown sequences with two individuals (only one being speaking at a time) are now used. The DE optimization method is

[1] The implementation of the DE algorithm is based on Storn's public domain software [9].

used to project the MFCCs on a new 1D subspace as defined in Sec. 2 using ΔECC as optimization criterion. The measure of the MI between the resulting audio feature vector F_A^{opt} and the video features of each mouth region allows to classify the mouth as "speaking" (highest value of MI) or "non-speaking" (lowest value of MI). The normalized difference of MI is always in favor of the active speaker, *i.e.* the correct speaking mouth region is always indicated (see Table 2).

Table 2. Normalized difference between the speaking and the non-speaking mouth regions' MI using the audio features optimized with the $-\Delta ECC$ cost function

Sequence	1	2	3	4	5
ΔI	84.23%	86.27%	95.55%	80.9%	76.15%

5 Conclusions

One central issue in the context of the multimodal speaker detection method described here is the optimization of an objective function based on MI. Since no approximation is made, neither of the pdf of the features (estimated from the samples), nor of the cost function, the optimization problem turns out to be a quite challenging one. The performances and limits of three optimization methods, the local Powell's method and the global GACS and DE, have been compared, showing that the intrinsic properties of the DE algorithm make it the best choice for the problem tackled here. As a result, the method is able to detect the current speaker on the five test sequences.

References

1. Butz, T., Thiran, J.P.: From error probability to information theoretic (multi-modal) signal processing. Signal Process. **85** (2005) 875–902
2. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. 2nd edn. Cambridge University Press (1992)
3. Schroeter, P., Vesin, J.M., Langenberger, T., Meuli, R.: Robust parameter estimation of intensity distributions for brain magnetic resonance images. IEEE Trans. Med. Imaging **17**(2) (1998) 172–186
4. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. J. Global Optim. **11** (1997) 341–359
5. Besson, P., Popovici, V., Vesin, J., Kunt, M.: Extraction of audio features specific to speech using information theory and differential evolution. EPFL-ITS Tech. Rep. 2005-018, EPFL, Lausanne, Switzerland (2005)
6. Besson, P., Kunt, M., Butz, T., Thiran, J.P.: A multimodal approach to extract optimized audio features for speaker detection. In: Proc. EUSIPCO. (2005)
7. Leung, Y.W., Wang, Y.: An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Trans. Evo. Comp. **5**(1) (2001) 41–53
8. Price, K.V.: 6: An Introduction to Differential Evolution. In: New Ideas in Optimization. McGraw-Hill (1999) 79–108
9. Storn, R.: Differential evolution homepage [online]. (Available: http://www.icsi.berkeley.edu/storn/code.html)

Artificial Life Models in Lung CTs

Sorin Cristian Cheran[1] and Gianfranco Gargano[2,*]

[1] Istituto Nazionale di Fisica Nucleare, Sezione di Torino,
Universita' degli Studi di Torino, Dipartimento di Informatica,
Associatione Sviluppo Piemonte
cheran@to.infn.it

[2] Istituto Nazionale di Fisica Nucleare, Sezione di Bari,
Universita' degli Studi di Bari, Dipartimento di Fisica
gianfranco.gargano@ba.infn.it

Abstract. With the present paper we introduce a new Computer Assisted Detection method for Lung Cancer in CT images. The algorithm is based on several sub-modules: 3D Region Growing, Active Contour And Shape Models, Centre of Maximal Balls, but the core of our approach are Biological Models of ants known as Artificial Life models. In the first step of the algorithm images undergo a 3D region growing procedure for identifying the ribs cage; then Active Contour Models are used in order to build a confined area for the incoming ants that are deployed to make clean and accurate reconstruction of the bronchial and vascular tree, which is removed from the image just before checking for nodules.

1 Introduction

The use of Chest Computer Tomography (CT) has lead to a better identification of lung cancer as well as the definition of its type, also reducing the number of benign nodules that are removed. The output of such an exam is a huge amount of data (series of 2D images) that in the end the radiologist must investigate and look upon. With the present paper we discuss a new CAD system that makes use of Artificial Life models and other supporting techniques. A flowchart would look as follows: region growing is used to reconstruct the rib cage, Active Contour models are bounding (?) the ribs cage. Ants are released in this newly created confined volume to reconstruct the bronchial and the vascular tree, and finally atfer the trees removal the hunt for nodules begins. In the next section some research work is introduced. The following section describes the Ant System and the paper closes with some conclusions and the discussion of future work.

2 Artificial Life

Artificial Life is the study of man-made systems which exhibit behaviors characteristic of natural living systems. In nature ants use *stigmergic* communication (by indirect

[*] On behalf of MAGIC-5 Collaboration.

interactions or by modifying the environment) using trails of pheromone [1]. These trails generally lead from the nest to different food-sources. When ants are trying to find a food source, generally the ant (trail C – Figure 1) that finds the shortest path reaches the nest quicker and thus its path has the most pheromone. Ants evolved the ability to cooperate, dividing their labor, such that all of them solve the tasks for which they are best suited in a robust manner.

2.1 Research Work

Among the pioneers of the field was Dorigo, that treated the Ant Colony Optimization problem [2] and showed how a group of ants can successfully find a close-to-optimal solution for the Traveling Salesman problem [3], Graph Coloring, Quadratic Assignment Problem [4] or the Job Shop scheduling.

Another pioneer of Collective was James Kennedy that, in 1995, proposed Particle Swarm Optimization (PSO) [5]. More related to the approach we pursue is the work by Chialvo and Millonas [6]. Based on the this paper, Ramos and Almeida [7] developed an extended model where ants are deployed in a digital habitat (image), such that the insects are able to move and to perceive it. In [8] Mazouzi and Batouche introduce a MAS (Multi-Agent System) for 3D object recognition. Other work can also be found in [9] where a cognitive MAS for the surveillance of dynamic scenes is discussed.

2.2 Materials

The Input of the algorithm are lung CT images, composed of a series of 2D files in DICOM format. The 2D image size is generally 512x512 pixels with a 16bits depth. The series is reconstructed as a 3D Matrix with a voxel size in the 3rd dimension which depends on the number of slices contained. The database for all the tests is provided by the MAGIC-5 Collaboration (add ref.). The images are being taken in the framework of the Regione Toscana (reference?) lung cancer screening program.

3 Ants

Ants are to be created and released in a confined 3-D world. Different algorithms could be developed. In the present paper we discuss the “Wander-approach” in which ants are randomly released in the habitat. While they wander about according to some well defined rules, they leave behind different quantities of pheromone. In the end a map of pheromone will be created that will represent the image that we are interested in: the reconstruction of the bronchial and vascular tree. Two kinds of ant individuals are present. The queen creates and “manages” all the ants, it does not move and does not perceive the habitat. The reconstructor is aware of the habitat and lives in it.

A third type, “the shaper”, that will try to recognize the nodules will soon be implemented.

The approach is based on an idea introduced by Ramos and Almeida in [7]. Let’s suppose that at time t an ant is in voxel k. At time $t + 1$ the ant is supposed to choose as next destination one of the 26th neighbours of k. This is done as follows: for each neighbour i the following probability is calculated:

$$P_{ik} = \frac{W(\sigma_i)w(\sigma_i)}{\sum_{j/k} W(\sigma_j)w(\sigma_j)} \tag{1}$$

where

$$W(\sigma) = \left(1 + \frac{\sigma}{1+\delta\sigma}\right)^{\beta} \tag{2}$$

is the function that depends on the σ pheromone density, β being the osmotropotaxic sensitivity and δ the sensory capacity, while $w(\Delta_i)$ is the probabilistic directional bias. The directional bias is a probability that each voxel receives when taking into account the initial direction of the ant (Fig. 1).

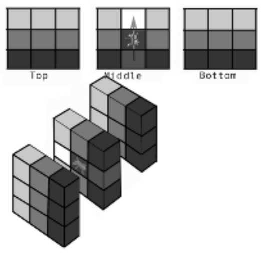


Fig. 1. One of the possible cases of directional bias. The lightest the colour of the voxel the higher the probability associated to that voxel.

As its next destination the ant will chose the voxel with the highest P_{ik}, and leaving a voxel k it will leave behind a certain amount of pheromone T:

$$T = \eta + p\theta_h \tag{3}$$

where η is the preset amount of pheromone, p is a constant and θ_h is the function that relates the amount of pheromone with the information provided by the image.

The deposited pheromone will evaporate with a certain rate and it will not diffuse in neighbouring voxels. For this paper (if not specified otherwise) θ_h is:

$$\theta_h = |I_i - I_k| \tag{4}$$

We used the gradient rule as the difference in intensities between two voxels situated on the different sides of the border of a branch will imply a big pheromone quantity being left behind by the visiting ants.

In this case the ant that is moving from voxel k to voxel i leaves behind an amount of pheromone equal to the gradient computed between the two voxels.

As a 2D image is a 3D image with just one slice the first tests were performed on 2D images. We started with different structures that we considered to be difficult, like

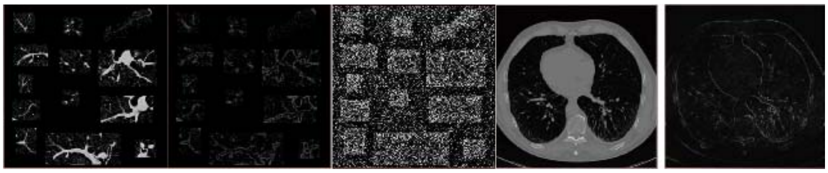


Fig. 2. First Image: Original structures extracted from a real CT. Second: the map of pheromones after 50 cycles. Third: the positions of the ants after 50 cycles. Fourth: Original CT, Fifth I: Pheromone Map for the CT slice.

branching points, round structures, intersection of branches, lung fissures, etc. In the examples from Fig. 2, θ_h was as described in **(4).**

The following step was to launch a colony of ants in full 2D CT image. As the image is 512x512 pixels we created a colony of 80.000 ants and started the algorithm. The output is represented in Fig. 3.

At this point we began 3D trials, by choosing different θ_h and in some cases the rules that we used made ants non-discriminative and the resulted reconstruction can contain all the lung one can see in Figure 3 (In this case the $\theta_h = k * I_i$).

Going to smaller structures, we tried regular "artificial" objects that the ants, using the "gradient rule" as θ_h, were able to fully reconstruct. This can be seen in Fig. 3.

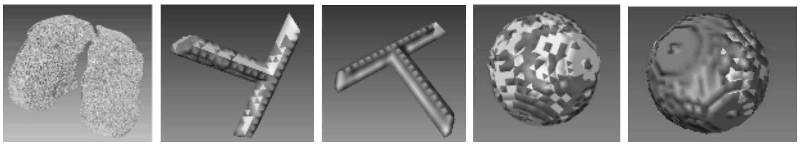


Fig. 3. Left: Zelous ants (reconstructing to much). Rest of the images: artificial branching point after 1 and 50 cycles, artificial sphere after 1 and 50 cycles.

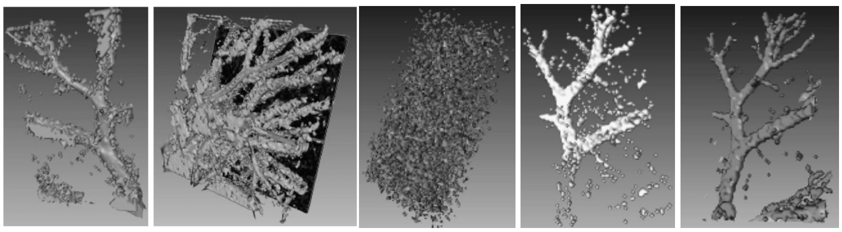


Fig. 4. Ants at work - depositing pheromone in the first and the second images. Third: . Reconstruction of 3D lung branch. Left: after 1 cycle. Forth : after 50 cycles, Fifth : after 100 cycles.

In Figure 4 (first 2 images) one can see the results of intermediate steps, when trying to reconstruct a small part of the vascular tree. After some testing one obtains the last 3 images in Fig. 4, which represent a part of the bronchial tree in the right lung. As one can see the reconstruction is not clean enough and further testing still needs to be done.

4 Conclusions

At the core of our algorithm stand Artificial life models (virtual ants or better, virtual termites if taken into account the existence of different types if individuals). One of our goals is to study and understand the collective behaviour of the ants in 3-D environments– worlds by making them capable of acknowledging the environment and of extracting useful information from it. We expect ants to behave differently when gaining a new grade of freedom which will lead to the revealing of new emergent behaviours of the colony based on ways the nature governs and controls our world. Another important aspect of the work is the creation of the CAD for lung CT. Once finished the reconstruction of the trees will be compared with already existing algorithms from image processing like 3-D region growing and the best performer will be included in the CAD (or one could chose a merging solution). We still have to test the algorithm on full CT images. This work will be done once the algorithm has been fully tested, with very good results, on smaller images.

References

1. Grasse P: "Termitologia, Tome II" Fondation des Sociétés. Construction. Paris; Masson, 1984.
2. Colorni A, Dorigo M, Maniezzo V : "Distributed Optimization by ant colonies" Proc. ECAL91, Paris, France, pp 134-142 Elsevier Publishing 1991.
3. Colorni A, Dorigo M, Maniezzo V: "The ant system: optimization by a colony of cooperat ing agents" IEEE Trans. Syst., Man. & Cybern. –PartB, vol 26, no.1, pp 1-13, 1996.
4. Colorni A, Dorigo M, Maniezzo V : "The Ant system applied to the quadratic assignment problem", Technical Report 94/28 of IRIDIA, Univeriste Libre de Bruxelles, 1994.
5. Eberhart R C, Kennedy J :" A New Optimizer Using Particles Swarm Theory", Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43, 1995
6. Chialvo D, Millonas M: "How Swarms Build Cognitive Maps". In Luc Steels (Ed.), The Biology and Technology of Intelligent Autonomous Agents, (144) pp. 439-450, NATO ASI Series, 1995.
7. Ramos V, Almeida F: "Artificial Ant Colonies in Digital Image Habitats" -A Mass Behav iour Effect Study on Pattern Recognition, Proceedings of ANTS´2000 -2nd International Workshop on Ant Algorithms (From Ant Colonies to Artificial Ants),
8. Mazouzi S, Batouche M: "A self-adaptive multi-agent system for 3D object recognition"
9. Remagnino P, Tan T, Baker K : "Multi-agent visual surveillance of dynamic scenes", Image and vision computing, Elsevier Ed., vol. 16, pp 529-532, 1998.

Learning High-Level Visual Concepts Using Attributed Primitives and Genetic Programming

Krzysztof Krawiec

Institute of Computing Science, Poznań University of Technology,
Piotrowo 2, Poznań 60965, Poland
krawiec@cs.put.poznan.pl

Abstract. In this paper, we present a novel approach to genetic learning of high-level visual concepts that works with sets of attributed visual primitives rather than with raster images. The paper presents the approach in detail and verifies it in an experiment concerning locating objects in real-world 3D scenes.

1 Motivations, Related Work, and Contributions

The primary motivation for research described in this paper is lack of a general methodology for design and development of pattern recognition systems. Manual design of such systems is for most real-world tasks tedious, time-consuming and expensive. The handcrafted solutions are usually limited in the scope of applicability and have poor ability to adapt, i.e., to perform *visual learning*.

In most approaches to visual learning reported in the literature, learning is limited to parameter optimization that usually concerns a particular processing step, such as image segmentation, feature extraction, etc. Only some methods [3,7,8,10,11,13] close the feedback loop of the learning process at the highest (e.g., recognition) level and test the proposed approach in a real-world setting. Reports on approaches that learn using raw images as training data, and, therefore, produce the entire object recognition system, are rather scant. Moreover, some of the proposed methods make use of domain-specific knowledge and are highly specialized towards a particular application.

This study on evolutionary visual learning is a step in a quest for more flexible, more universal, and more effective *representations* of pictorial information and the methods of its processing. In our former work on feature synthesis [1,5] we proposed a visual learning framework inspired by linear genetic programming and tested it on different object recognition tasks. Despite encouraging results, we came to the conclusion that further progress cannot be made without referring to representations of visual information other than raster images, and without providing more flexible mechanisms of its processing. Contributions of this paper address these issues and may be shortly characterized as follows. Firstly, the proposed approach works with *visual primitives* derived from input the image rather than with raster images. Secondly, for this purpose, we develop a novel variant of typed genetic programming (GP) customized to process such visual primitives.

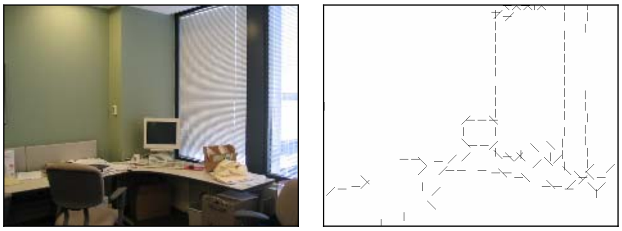


Fig. 1. An exemplary image (left) and its corresponding VP representation (right)

2 The Proposed Approach: Learning Based on Visual Primitives

The main novelty of the proposed approach consists in *abstracting from raster images* and using *visual primitives* (VPs) as basic granules of visual information. Prior to the learning process, the input image undergoes an appropriate preprocessing that leads to its representation in terms of visual primitives (VPR for short). Each VP is described by four scalar *attributes*: x coordinate, y coordinate, orientation, and intensity. We employ *Gabor filters* [2] with four different orientations α (0°, 45°, 90°, and 135°) to extract VPs from the monochrome input image. A local maximum at location (x,y) in α-Gabor filter response produces a VP $p = (x, y, \alpha)$. To limit the number of VPs in VPR, only a predefined β-percentile of the brightest pixels in Gabor filter responses are taken into account. Also, no two VPs with the same orientation can be located closer than some predefined distance d_{min}. The resulting VPR is, a *set* of attributed VPs, is usually several orders of magnitude more compact than the original image. Figure 1 presents an exemplary image and its corresponding VP representation for $\beta = 0.05$ and $d_{min} = 4$. Each segment in VPR depicts a single VP, with its (x, y) coordinates located in the middle of the segment and orientation depicted by slant.

The visual learning in the proposed approach employs a variant of GP [4]. To meet the *principle of least commitment* [9], we allow the GP expressions/trees to work directly with sets of VPs rather than only with some predefined scalar features derived from them. To this aim, a *typed* GP is used with the following three basic types: scalars ($\Re$ for short), sets of VPs (P for short), and attribute labels (A for short). Consequently, there are three types of terminal GP nodes, which yield, respectively, an ephemeral random constant, a VPR of the input image z, and a tag of primitive attribute: *Coordinate_X*, *Coordinate_Y*, *Orientation,* or *Intensity*. The non-terminal GP operators may be divided into three following categories. *Scalar operators* accept and return arguments of type $\Re$ (as in standard GP applied to symbolic regression; see [4]). *Selectors* accept at least one argument of type P and return result of type P. A selector filters out *some* of the VPs it receives from its child node(s) according to some criterion/condition. *Aggregators* accept at least one argument of type P and return result of type $\Re$. An *aggregator* combines, according to some rule, the values of a chosen primitive attribute of *all* the VPs it receives from its child node(s).

Scalar operators encompass the basic arithmetic and fundamental functions (see Table 1). For both selectors and aggregators, there are two variants of them: parametric and non-parametric. *Non-parametric selectors* expect two child nodes of

Table 1. The complete list of GP operators

Type	*Non-parametric*	*Parametric*
Scalar	+, –, ×, /, *sin*, *cos*, *exp*, *log*	
Selector	∪, ∩, \, *SymmetricDifference*	*Equals*, *LessThan*, *GreaterThan*
Aggregator	*Sum*, *Mean*, *Product*, *Median*	*Moment*, *CentralMoment*, *Percentile*

type P and produce an output of type P. Operators that implement basic set algebra, like set union, intersection, or difference, belong to this category. *Parametric selectors* expect three child nodes of types P, A, and $\Re$, and produce an output of type P. For instance, the operator *LessThan* applied to child nodes (P, *Orientation*, 70) filters out all VPs from P for which the value of the attribute *Orientation* is less than 70. *Non-parametric aggregators* expect two child nodes of types P and A, and produce an output of type $\Re$. Operators of this type implement mostly statistical descriptors, e.g., the operator *Mean* applied to child nodes (P, *Coordinate_X*) computes the mean value of coordinate x for all VPs in P. *Parametric aggregators* expect three child nodes of types P, A, and $\Re$, and produce an output of type $\Re$. For instance, the operator *CentralMoment* applied to child nodes (P, *Coordinate_Y*, 2) computes the central moment of order 2 of coordinate y for all VPs in P. Table 1 presents the complete list of GP operators used in the computational experiments.

3 Experimental Results

For experimental verification, we chose the task of *locating* computer screens in 38 images taken from the 'aug1_static_atb_office_bldg400' folder of the *MIT-CSAIL Database of Objects and Scenes* [14]. The images exhibit significant variability of brightness and proximity of monitor case, the state of the screen (on/off), lighting conditions, and scene contents (see example in Fig. 1). To avoid bias toward the center of the scene where the screens are most often to be found, we created extra examples by cropping each training example from all sides, what lead to (1+4)×38 = 190 examples. The original images have been converted to grayscale and downscaled to 128×96 pixels. Next, basing on the preprocessed image, we create its VPR using the procedure described in Section 3. For the resulting training set of images I, the number of VPs varied from 103 to 145 per image (122.2 on the average).

The experiment consisted in evolving a population of 1000 individuals for 50 generations, using a minimized fitness function f defined as follows:

$$f(s) = \sum_{z \in I} \sum_{p \in s(z)} d(p) / |I|, \tag{1}$$

where $s(z)$ denotes the set of visual primitives produced by an individual s given image z as an input, and $d(p)$ denotes the Euclidean distance between the VP p and the actual screen position. Most of GP-related parameters are set to their defaults used in ECJ library [6]. This includes: algorithm type (generational), mutation probability

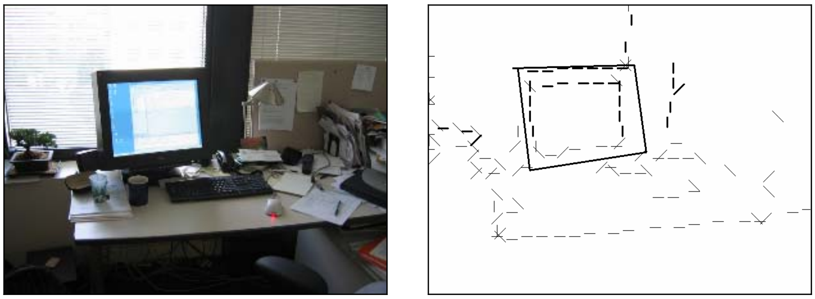


Fig. 2. An input image (left) and the result produced by the best evolved individual (right)

```
(SelectorGreaterThan
   (SelectorLessThan
      (SelectorGreaterThan
         (SelectorLessThan
            (SelectorGreaterThan
               VPR
               Coordinate_Y
               (exp 0.96645296))
            Orientation
            (exp 0.96645296))
         Coordinate_X
         (cos
            (Median
               (SelectorGreaterThan
                  VPR
                  Intensity
                  0.18945523)
               Orientation)))
      (continued in right column)
```

```
         Coordinate_Y
         (+
            (-
               (% (% 0.9664 -0.5619) 0.8692)
               -0.8356722)
            (Mean
               (SelectorGreaterThan
                  VPR
                  Intensity
                  (exp 0.96645296))
               Coordinate_Y)))
      Intensity
      (exp
         (-
            (-
               (cos
                  (Median VPR Orientation))
               -0.8356722)
            -0.8356722)))
```

Fig. 3. The textual representation of the best individual found during the evolutionary run

(0.1), mutation type (one-point), crossover probability 0.9, maximum tree depth allowed for individuals modified by genetic operators: 7, number of retries if the resulting individual does not meet this limit: 3. The procedure selecting the tree nodes for mutation or crossover selects internal tree nodes with probability 0.9, and tree leaves with probability 0.1. The software used ECJ [6]) and JAI [12] libraries.

Figure 2 shows the result of recognition performed by the best individual found during the evolutionary run. In the VP image part, the closed polygon shows the actual location of the recognized object (computer screen). The short thick segments depict the VPs selected by the individual, whereas the thin segments correspond to those VPs which were originally present in the VPR of the input image, but have been filtered out by the individual. In Fig. 3, we present the code of the best GP individual evolved in the evolutionary run.

4 Conclusions

The major feature of the proposed approach is the abstraction from raw raster images, which enables the learning process to develop advanced visual concepts that may successfully recognize complex objects in real-world views of 3D scenes. The approach proceeds by selecting subsets of VPs and aggregating their attribute values; this idea is general and may be easily adapted to any kind of set-based representation using attributed visual primitives. Note also that the VP representation is both *application- and task-independent*. There is virtually no domain-specific knowledge that would bias the method towards the considered task of monitor screen location.

The compactness of VP representation reduces significantly the computational complexity as compared to processing of raster images. In our experiment, the time of individual evaluation amounted to 84ms on the average (Pentium 3.0 GHz). Moreover, as the original VP representation of a particular image does not change during evolution, it may be computed once, which reduces the learning time even more.

Acknowledgments

This research has been supported by KBN research grant 3 T11C 050 26.

References

1. Bhanu, B., Lin, Y., Krawiec, K.: Evolutionary Synthesis of Pattern Recognition Systems. Springer-Verlag, Berlin Heidelberg New York (2005)
2. Gabor, D.: Theory of Communication, J. Inst. of Electrical Engineers 93 (1946) 429–457
3. Johnson, M.P., Maes, P., Darrell, T.: Evolving visual routines. In: Brooks, R.A., Maes, P. (eds.): Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems. MIT Press, Cambridge, MA (1994) 373–390
4. Koza, J.R., Andre, D., Bennett III, F.H., Keane, M.A.: Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufman, San Francisco, CA (1999)
5. Krawiec, K., Bhanu, B.: Visual Learning by Coevolutionary Feature Synthesis. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 35 (2005) 409–425
6. Luke, S.: ECJ Evolutionary Computation System. http://www.cs.umd.edu/projects/plus/ec/ecj/ (2002)
7. Maloof, M.A., Langley, P., Binford, T.O., Nevatia, R., Sage, S.: Improved rooftop detection in aerial images with machine learning. Machine Learning 53 (2003) 157–191
8. Marek, A., Smart, W.D., and Martin, M.C.: Learning Visual Feature Detectors for Obstacle Avoidance using Genetic Programming. In: Late Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2002), New York (2002) 330–336
9. Marr, D.: Vision. W.H. Freeman, San Francisco, CA (1982)
10. Rizki, M., Zmuda, M., Tamburino, L.: Evolving pattern recognition systems. IEEE Transactions on Evolutionary Computation 6 (2002) 594–609
11. Song, A., Ciesielski, V.: Fast texture segmentation using genetic programming. In: Sarker, R., et al. (eds.): Proceedings of the 2003 Congress on Evolutionary Computation CEC2003. IEEE Press, Canberra (2003) 2126–2133
12. Sun Microsystems Inc.: Java Advanced Imaging API Specification. Version 1.2 (2001)
13. Teller, A., Veloso, M.M.: PADO: A new learning architecture for object recognition. In: Ikeuchi, K., Veloso, M. (eds.): Symbolic Visual Learning. Oxford Press, New York (1997) 77–112
14. Torralba, A., Murphy, K.M., Freeman, W.T.: MIT-CSAIL Computer vision annotated image library. http://web.mit.edu/torralba/www/database.html (2004)

Evolutionary Denoising Based on an Estimation of Hölder Exponents with Oscillations

Pierrick Legrand[1,2], Evelyne Lutton[2], and Gustavo Olague[1]

[1] CICESE, Research Center, Applied Physics Division,
Centro de Investigación Científica y de,
Educación Superior de Ensenada, B.C.,
Km. 107 carretera Tijuana-Ensenada,
Ensenada, 22860, B.C. México
[2] INRIA Rocquencourt, Complex Team,
Domaine de Voluceau BP 105,
Le Chesnay Cedex 78153, France

Abstract. In multifractal denoising techniques, the acuracy of the Hölder exponents estimations is crucial for the quality of the outputs. In continuity with the method described in [1], where a wavelet decomposition was used, we investigate the use of another Hölder exponent estimation technique, based on the analysis of the local "oscillations" of the signal. The associated inverse problem to be solved, i.e. finding the signal which is the closest to the initial noisy one but having the prescribed regularity, is then more complex. Moreover, the associated search space is of a different nature as in [1], which necessitates the design of ad-hoc genetic operators.

1 Introduction

In the past years many different signal and image denoising techniques have been proposed, some of them being even based on artificial evolution [1, 2]. The basic notations are the following. One observes a signal or an image Y which is some combination $F(X, B)$ of the signal of interest X and a noise B. Making various assumptions on the noise, the structure of X and the function F, one then tries to obtain an estimate $\hat{X}$ of the original signal, optimal in some sense. We consider denoising as equivalent to increasing the Hölder function α_Y (see section 2 for definitions) of the observations. Indeed, it is generally true that the local regularity of the noisy observations is smaller than the one of the original image, so that in any case, α_X should be greater than α_Y.

In this paper, section 2 recalls some basic facts about Hölder regularity analysis. We describe in section 3 how oscillations are used to provide an estimator of the Hölderian regularity. The new denoising method is explained in section 4 and the evolutionary algorithm, with its ad-hoc genetic operators, are detailed in section 5. Numerical experiments are presented in section 6.

2 Hölder Regularity

To simplify notations, we deal with 1D signals, and we assume that signals are nowhere differentiable. Generalisation to differentiable signals simply requires to introduce polynomials in the definitions [3]. Below the definitions of the pointwise and local Hölder exponents are given.

Let $\alpha \in (0,1)$, and $x_0 \in K \subset R$. A function $f : K \rightarrow R$ is in $C^{\alpha}_{x_0}$ if for all x in a neighbourhood of x_0, $|f(x) - f(x_0)| \leq c|x - x_0|^{\alpha}$ (2) where c is a constant. **The pointwise Hölder exponent** of f at x_0, denoted $\alpha_p(f, x_0)$, is the supremum of the α for which (2) holds.

Let us now introduce the local Hölder exponent: Let $\alpha \in (0,1)$, $\Gamma \in R$. One says that $f \in C^{\alpha}_{l}(\Gamma)$ if $\exists C : \forall x, y \in \Gamma : \frac{|f(x)-f(y)|}{|x-y|^{\alpha}} \leq C$ (3). Let $\alpha_l(f, x_0, \rho) = \sup\{\alpha : f \in C^{\alpha}_{l}(B(x0, \rho))\}$. **The local Hölder exponent** of f at x_0 is $\alpha_l(f, x_0) = \underset{\rho \rightarrow 0}{lim} \alpha_l(f, x_0, \rho)$.

Since α_p and α_l are defined at each point, we may associate to f two functions $x \rightarrow \alpha_p(f, x)$ and $x \rightarrow \alpha_l(f, x)$ which are two different ways of measuring the evolution of its regularity.

The quality of a denoising technique based on these exponents, strongly relies on the quality of an estimator of these quantities. In [1], the estimation was performed by a wavelet technique. We will see in the sequel that a better estimation of the Hölder exponent can be obtained by measuring the oscillations of the function.

3 Estimation by Oscillations

The estimation based on oscillations measurements is a direct application of the local Hölder exponent definition (see [4]). The condition (3) can be written as: A function $f(t)$ is Hölderian with exponent $\alpha \in [0,1]$ at t if there exists a constant c, for all τ: $osc_{\tau}(t) \leq c\tau^{\alpha}$ with

$$osc_{\tau}(t) = \underset{|t-t'|\leq\tau}{sup} f(t') - \underset{|t-t'|\leq\tau}{inf} f(t') = \underset{t',t''\in[t-\tau,t+\tau]}{sup} |f(t') - f(t'')|.$$

At each point we estimate the pointwise Hölder exponent as the slope of the regression of the logarithm of the oscillation versus the size of the window τ. As

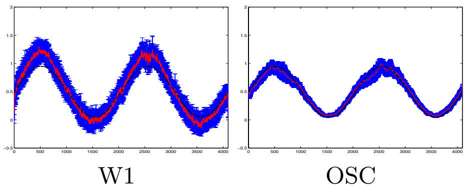


Fig. 1. 10 multifractional Brownian Motions have been built with a regularity H evolving like a sine. The 2 methods of estimation of the Hölderian regularity have been applied: a wavelet-based (W1) and the method by oscillations (OSC). After an optimisation of the parameters of the 2 methods in term of risk, the means of the estimated Hölder functions are displayed.

we see in figure 1, the estimation by oscillations provides better results than an estimation by wavelets.

4 Method

According to the notation of section 1, we seek a denoised version $\hat{X}$ of the observed signal Y that meets the following constraints:

1) $\hat{X}$ is close to Y in the L^2 sense.
2) The Hölder function of $\hat{X}$ is prescribed.

If α_X is known, we choose $\alpha_{\hat{X}} = \alpha_X$. In some situations, α_X is not known but can be estimated from Y, see [5]. Otherwise, we just set $\alpha_{\hat{X}} = \alpha_Y + \delta$, where δ is a user-defined positive function, so that the regularity of $\hat{X}$ will be everywhere larger than the one of the observations. Two problems have to be solved in order to obtain $\hat{X}$. First, a procedure that computes the Hölder function of a signal from discrete observations is needed. Second, we need to be able to manipulate the data so as to get a specific regularity. To solve the first problem, the estimation method of section 3 is used, and for the second problem, an evolutionary algorithm has been designed.

5 Evolutionary Algorithm

We consider that an individual is a signal (1D or 2D). On the contrary to [1] where an individual was made of a subset of wavelet coefficients, a direct encoding of the signal in the genome has been used.

Initialisation: As the search space is extremely large, a direct search starting from a random set of initial signals has no chance to provide a good denoising in a reasonable time. However, many initial guess are available, including the noisy signal itself. We actually use several deterministic denoising methods to provide the initial population. These methods are the **Multifractal Bayesian Denoising**[6] and the **Multifractal Pumping**[5], and depend from a parameter setting. The parameters are generated randomly.
Fitness: An important point of the method is that the fitness calculation is based on two kind of fitness function. A pointwise fitness has been defined for each point of the signal as a combination between the quality of the individual in term of regularity and in term of distance to the noisy signal. As said above, it is based on the estimation by oscillations. The pointwise fitness is then combined to provide a local fitness. The local fitness is the sum of the pointwise fitness on a given segment (or window). The local fitness is used in the crossover and in the mutation operators. We compute the global fitness when we perform this sum on the full signal. This fitness is used for the selection and for the ranking.
Crossover: A simple ranking selection mechanism with selective pressure 2 is used to select two individuals. Random crossing points are then selected. For images, a set of random rows and columns is chosen. The local fitness on each

resulting segment is then used to select the best parts of the two individuals as the corresponding segment of the child.
Mutation: In a similar way, each segment (or image window) is muted using a probability law inversely proportional to the local fitness. For each individual we consider the worst local fitness wlf i.e. the fitness of the worst segment. Let $lf(j,i)$ the local fitness of the i^{th} segment of the j^{th} individual. The probability of mutation for this segment is $Pm(j,i) = \frac{lf(j,i)}{wlf}$.

6 Numerical Results

For the first example (see figure 2), the original signal is a Generalized Weierstrass function with a regularity $\alpha(X,t) = t$ with $t \in [0,1]$. This signal is corrupted by a white Gaussian noise (standard deviation equal to 0.3). We use a synthetic image to perform an experiment in 2 dimensions. The figure 3 shows

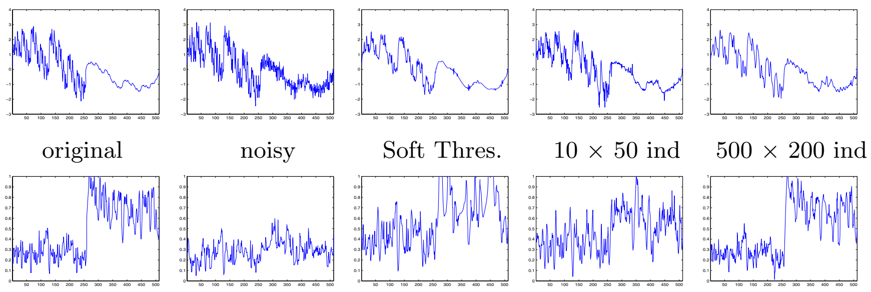


Fig. 2. First row: original Generalized Weierstrass Function, noisy version, denoising with Soft Thresholding, denoising by our method after 10 generations and 50 individuals, denoising by our method after 500 generations and 200 individuals. Second row: corresponding Hölder functions. Our method allow to recover almost perfectly the Hölder function of the original signal.

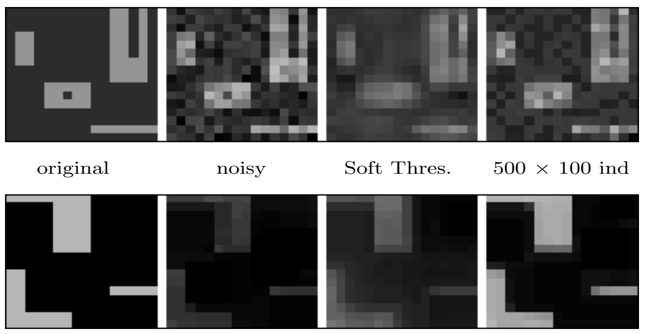


Fig. 3. Original image, the noisy one, a denoising by Soft Thresholding and by our method (100 ind, 500 gen). The second row displays the corresponding Hölder functions.

the original image, the noisy one, a denoising by Soft Thresholding and by our method. The second row displays the corresponding Hölder functions. As in the previous examples, our method allows to obtain a denoised version of the signal with the prescribed regularity.

7 Conclusion

We have experimented in this paper a new scheme for a multifractal denoising technique. It is based on a more precise and more complexcomputation of the Hölder exponent of a signal. This work is actually a first attempt to use an estimation of Hölder exponent based on oscillations for signal enhancement. Preliminary experiments yield satisfactory results, with a more precise control of the reconstructed regularity, which has to be considered as a major advantage for this type of techniques. Moreover, the evolutionary engine that has been designed has the following interesting characteristics: it performs a basic hybridisation with two other denoising techniques (Multifractal Bayesian Denoising and Multifractal Pumping for the initialisation step), and uses locally optimised genetic operators. Further work will first consist in a more precise analysis of the locally optimised genetic operators in comparison with classical "blind" ones. Second, the hybridisation scheme has to be investigated as it may be a good solution to reduce computation costs of the method. Additionally, the availability of a pointwise and local definition of the fitness opens the way to "Parisian" evolution implementations for the genetic engine. This may be another solution to reduce computational expenses of the method, see for example [7, 8].

References

1. J. Levy Vehel and E. Lutton, "Evolutionary signal enhancement based on Hölder regularity analysis," *EVOIASP2001, LNCS 2038*, 2001.
2. Pierre Grenier, Jacques Levy Vehel, and Evelyne Lutton, "An interactive ea for mulifractal bayesian denoising," *EvoIASP 2005, 30 March - 1 April, Lausanne*, 2005.
3. Y. Meyer, "Wavelets, Vibrations and Scalings," *American Mathematical Society, CRM Monograph Series*, vol. 9, 1997.
4. C. Tricot, *Curves and Fractal Dimension*, Springer-Verlag, 1995.
5. P. Legrand, "Débruitage et interpolation par analyse de la régularité hölderienne. application à la modélisation du frottement pneumatique-chaussée, phd thesis," *Université de Nantes et Ecole Centrale de Nantes*, 2004.
6. J. Levy Vehel and P. Legrand, "Bayesian multifractal denoising," *ICASSP 2003, IEEE internat. conf. on Acoustics, Speech, and Signal Processing, Hong Kong*, April 6-10 2003.
7. Frederic Raynal, Pierre Collet, Evelyne Lutton, and Marc Schoenauer, "Polar ifs + parisian genetic programming = efficient ifs inverse problem solving," *Genetic Programming and Evolvable Machines Journal, Volume 1, Issue 4, pp. 339-361, October*, 2000.
8. Enrique Dunn, Gustavo Olague, and Evelyne Lutton, "Automated photogrammetric network design using the parisian approach," *EvoIASP 2005, 30 March - 1 April, Lausanne*, 2005.

Probability Evolutionary Algorithm Based Human Body Tracking

Shuhan Shen and Weirong Chen

School of Electrical Engineering, Southwest Jiaotong University,
Chengdu, China
sh.shen@163.com

Abstract. A novel evolutionary algorithm called Probability Evolutionary Algorithm (PEA), and a method based on PEA for visual tracking of human body are presented. PEA is inspired by the Quantum computation and the Quantum-inspired Evolutionary Algorithm, and it has a good balance between exploration and exploitation with very fast computation speed. In the PEA based human tracking framework, tracking is considered to be a function optimization problem, so the aim is to optimize the matching function between the model and the image observation. Then PEA is used to optimize the matching function. Experiments on synthetic and real image sequences of human motion demonstrate the effectiveness, significance and computation efficiency of the proposed human tracking method.

1 Introduction

With the fast developments of computer science and technology, visual analysis of human motion in image sequences interests more and more researchers from both laboratory and industry. Human tracking is a particularly important issue in human motion analysis and it has been a popular topic in the research of computer vision.

Tracking can be divided into region-based, feature-based, active-counter-based and model-based tracking [1]. Model-based tracking can provide abundant informa tion of human motion, but the increasing of subparts of the human model would potentially incur high dimensionality and make tracking a difficult task. Different from using particle filters within the Bayesian framework [2-6], human tracking is considered to be a function optimization problem in this paper, so the aim is to optimize the matching function between the model and the observation. Function optimization is a typical application area of Genetic Algorithms (GAs), but canonical genetic algorithms is hard to be used here due to the high dimensionality of human model and the requirement of computation speed. In this paper, we present a novel evolutionary algorithm called Probability Evolutionary Algorithm (PEA) which is inspired by the Quantum computation [7] and Quantum-inspired Evolutionary Algorithm (QEA) [8], and then the PEA based human body tracking is proposed in which PEA is used to optimize the matching function. PEA has a good balance between exploration and exploitation with very fast computation speed, and it is suitable for human tracking and other real-time optimization problems.

2 PEA

PEA is characterized by probabilistic superposed bit, observation and update method. In PEA, the individual is encoded by probabilistic superposed bit, which is defined as the smallest unit of information in PEA, as below:

$$\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_k \end{bmatrix}, p_0 + p_1 + \cdots + p_k = 1 \tag{1}$$

where P_0, P_1, … P_k give the probability that a probabilistic superposed bit will be observed in the '0' state, the '1' state, …, and the '*k*' state, respectively. So a probabilistic superposed bit is a linear superposition of the states 0 to *k*. In PEA, an individual is defined as a string of probabilistic superposed bits, so the individual is no longer a deterministic state, but a linear superposition of all kinds of states.

For the individual can't be used in the fitness function directly, an observation step is used to get the observed individual which is a deterministic *k*-nary string.

The update operator is the only evolutionary operator in PEA which can increase the observation probabilities of some states, and decrease the observation probabilities of some other states, in order to make the high fitness state be observed more likely.

The details of the observation and the update step, and the procedure of PEA could be found in our former paper [9].

3 PEA Based Human Tracking

3.1 The Framework of PEA Based Human Tracking

Different from tracking human using particle filters within the Bayesian framework, tracking is considered to be a function optimization problem in this paper. We denote the human model by **X**, and denote the observation associate with **X** by **Z**. The function f (**X**,**Z**) represents the matching degree between **X** and **Z**. Assume that we have known that the model at time instance t-1 is X^{t-1}, so the model $\mathbf{X}^t$ at time instance t can be get by equation 2.

$$\mathbf{X}^t = \mathbf{X}^{t-1} + \Delta\mathbf{X} \tag{2}$$

Here, $\Delta\mathbf{X}$ is the change of the model $\mathbf{X}^{t-1}$. After we get $\mathbf{X}^t$, the matching function $f(\mathbf{X}^t,\mathbf{Z}^t)$ can be calculated. Since $\mathbf{X}^t$ is associated with $\Delta\mathbf{X}$, the matching function can be written as:

$$f(\mathbf{X}^t,\mathbf{Z}^t) = g(\Delta\mathbf{X}) \tag{3}$$

So tracking at time instance t is to optimize g ($\Delta\mathbf{X}$) in $\Delta\mathbf{X}$'s search space. Generally, g ($\Delta\mathbf{X}$) is a multi-modal function with many local best solutions, and conventional optimization methods are difficult to get the global best solution, so we use PEA to optimize g ($\Delta\mathbf{X}$).

3.2 Human Model

We employ a 10-part articulated human body model which consists of 10 parts and each pairs of neighbor parts are connected by the joint point, as shown in Fig. 1.

The model has 10 joints, and the root joint is at the middle bottom of the trunk. The root joint has 3 degrees, and each of the other 9 joints has 1 degree. The model **X** can be written as:

$$\mathbf{X} = \{x, y, \theta_1, \theta_2, \ldots\theta_{10}\} \tag{4}$$

Here, x and y represent the location of the root joint, and $\theta_1,\theta_2\ldots\theta_{10}$ represent the swiveling angles of the 10 joints. $\Delta\mathbf{X}$ can be written as:

$$\Delta\mathbf{X}=\{\Delta x, \Delta y, \Delta\theta_1, \ldots\Delta\theta_{10}\} \tag{5}$$

Human motion is a gradually changed movement, so ΔX can be limited in a logical small scope. This scope can be learned or man-made.

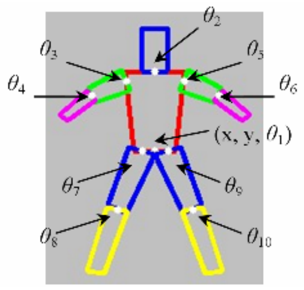


Fig. 1. 2D human body model

4 Experimental Results

Two image sequences are used here to demonstrate the effectiveness of PEA. Sequence 1 is a synthetic image sequence generated by Pose software [10] which consists of 100 frames. Sequences 2 is a real image sequence which consists of 325 frames. The observation **Z** is also an important factor in tracking. Here we use two types of visual cues: edge and intensity. We compared the tracking results from PEA with Annealed Particle Filtering (APF). All the algorithms run on a 2.4GHz PC without code optimization.

4.1 Parameters Setting

In APF based tracking, 200 particles are used, and the particles are annealed for 8 times. In PEA based tracking, the population size is set to 4, and the maximum number of generations is 200.

4.2 Results

Some tracking results of PEA and APF for sequence 1 and sequence 2 are shown in Fig. 2 and Fig. 3 respectively. The average computation time for one frame of PEA

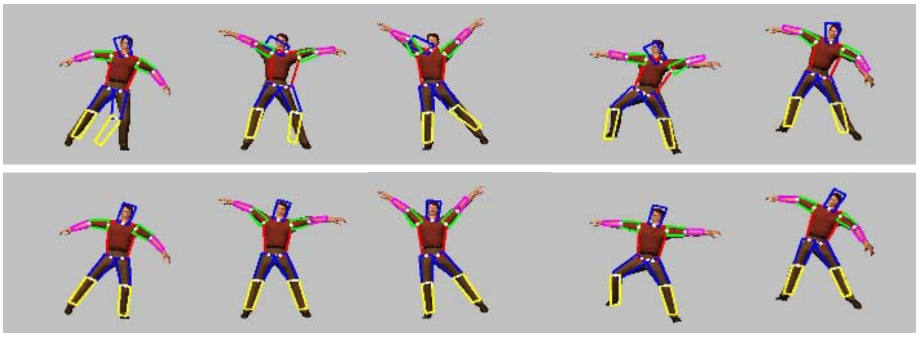


Fig. 2. Some tracking results of sequence 1. The top row is the tracking results based on APF. The bottom row is the tracking results based on PEA.

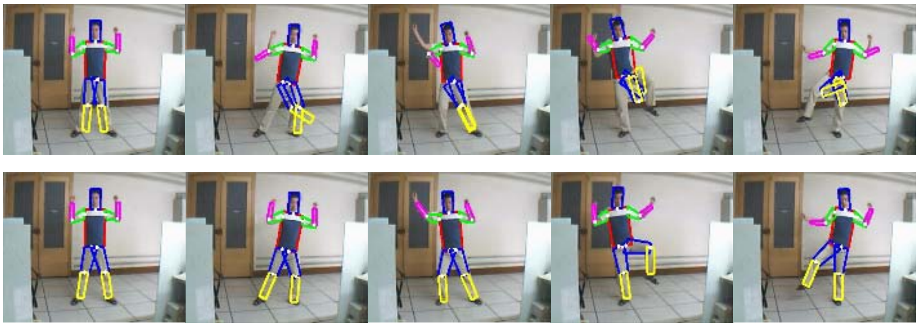


Fig. 3. Some tracking results of sequence 2. The top row is the tracking results based on APF. The bottom row is the tracking results based on PEA.

Table 1. Average computation time for one frame

Algorithm	Particles or Population size	second/frame
APF	200	3.77s
PEA	4	1.62s

and APF are shown in Table 1. The results show that the PEA based tracking algorithm yields more stable results than APF, and run much faster than APF.

In the experiments we also found that, when the population size is bigger than 10, the tracking result can not be improved further, so we suggest that the population size is set to 2 to 8 in applications in order to get a balance between the tracking accuracy and the computation time.

5 Conclusions

Model-based human tracking is a challenging problem, since the human model has high dimensionality. Different from tracking human using particle filters, we consider tracking to be a function optimization problem, and a novel evolutionary algorithm called Probabilistic Evolutionary Algorithm (PEA) is proposed to optimize the matching function between the model and the observation. PEA has a good balance between exploration and exploitation with very fast computation speed. Experiments on synthetic and real image sequences of human motion demonstrate the effectiveness, significance and computation efficiency of the PEA based human body tracking algorithm.

References

1. Hu, W.M., Tan, T.N., Wang, L., Maybank, S.J.: A survey on visual surveillance of object motion and behaviors. IEEE Trans. on System Man and Cybernetics 34 (2004) 334–351
2. Gavrila, D., Davis, L.: 3D model based tracking of humans inaction: A multiview approach. In: IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition, San Francisco, California (1996) 73–80.
3. Isard, M., Blake, A.: CONDENSATION-conditional density propagation for visual tracking. International Journal of Computer Vision 29 (1998) 5–28
4. Deutscher, J., Davidson, A., Reid, I.: Articulated partitioning of high dimensional search spaces associated with articulated body motion capture. In: IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition, Hawaii (2001) 669–676
5. Wu, Y., Hua, G., Yu, T.: Tracking Articulated Body by Dynamic Markov Network. In: Proceedings of the Ninth IEEE International Conference on Computer Vision (2003) 1096–1101
6. Zhao, T., Nevatia, R.: Tracking Multiple Humans in Crowded Environment. In: IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (2004) 342–349
7. Han, K.H., Kim, J.H.: Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization. IEEE Trans. on Evolutionary Computing 6 (2002) 580–593
8. Hey, T.: Quantum computing: An introduction. Computing & Control Engineering Journal 10 (1996) 105–121
9. Shen, S.H., Jiang, W.K., Chen, W.R.: Research of Probability Evolutionary Algorithm. In: 8th International Conference for Young Computer Scientists, Beijing (2005) 93-97
10. Poser Software: Available from http://www.curiouslabs.com

On Interactive Evolution Strategies

Ron Breukelaar[1], Michael Emmerich[1], and Thomas Bäck[1,2]

[1] Natural Computing Group, University of Leiden, Niels Bohrweg 1, Leiden 2333-CA, The Netherlands
{baeck, rbreukel, emmerich}@liacs.nl
http://www.liacs.nl

[2] NuTech Solutions GmbH, Martin Schmeisser Weg 15, Dortmund 44227, Germany

Abstract. In this paper we discuss Evolution Strategies within the context of interactive optimization. Different modes of interaction will be classified and compared. A focus will be on the suitability of the approach in cases, where the selection of individuals is done by a human user based on subjective evaluation. We compare the convergence dynamics of different approaches and discuss typical patterns of user interactions observed in empirical studies.

The discussion of empirical results will be based on a survey conducted via the world wide web. A color (pattern) redesign problems from literature will be adopted and extended. The simplicity of the chosen problems allowed us to let a larger number of people participate in our study. The amount of data collected makes it possible to add statistical support to our hypothesis about the performance and behavior of different Interactive Evolution Strategies and to figure out high-performing instantiations of the approach.

The behavior of the user was also compared to a deterministic selection of the best individual by the computer. This allowed us to figure out how much the convergence speed is affected by noise and to estimate the potential for accelerating the algorithm by means of advanced user interaction schemes.

1 Introduction

The research field of human-algorithm interaction (HAI) puts forward the involvement of humans in algorithmic solution processes. In contrast to human computer interaction the focus of this technology is on computational processes that are assisted by users. In contrast to interactive software like text processing systems or drawing software, the main structure of the solution process for the higher level task is still governed by the algorithm. The user has to assist the algorithm at some stages, that call for decisions based on subjective preferences, or that require the insights of experts in a problem, the formalization of which is often very difficult.

On a very global level we propose to distinguish between *reactive* or *proactive* interaction, i.e. user feedback requested by the algorithm, or optional interventions by the users into an autonomously running algorithm. An example for reactive feedback could be the request of an optimization algorithm on the subjective evaluation of solutions by means of the user. Contrarily, an example for proactive feedback would be given, if the user halts an optimization algorithm that is in a phase of stagnation, changes some parameters, and lets it continue with the changed settings. A boundary case for proactive feedback would be, if the user simply decides to finish an algorithm and pushes some 'stop' button that terminates it.

Among the few algorithm classes that already integrate the user in the computational process, interactive Evolutionary Algorithms (EA) are one of the most well known. Applications range from arts [1] and music [11], to industrial engineering applications [12, 7], mixture optimization [10], and prototyping in product design [4]. An excellent overview on applications of IEA was given by Banzhaf [4] and more recently by Takagi et al. [17].

In this paper we will mainly focus on the discussion of interactive variants of Evolution Strategies (ES) [14, 16]. ES are instantiations of Evolutionary Algorithms that are mainly used for the purpose of parameter optimization. In particular they feature the self-adaptation of step-size parameters. This allows to minimize the effort of the user as for many other EA the choice of the adequate parameters can cause a significant problem for the unexperienced user. Moreover, the self-adaptation makes it possible to automatically scale the behavior of the variation operator between a more exploratory coarse sampling or a finer sampling, which is needed to achieve a high accuracy to the end of the optimization.

ES have been already successfully applied for interactive optimization in parametric design. In particular the pioneering work of Herdy [9, 10] in this field should be mentioned here, who applied interactive variants of the ES to various problems ranging from the design of color mixtures to the search for coffee mixtures that meet a desired taste.

However, we belief that there are still many open questions with regard to Interactive Evolution Strategies (IES). For example step-size adaptation deserves further attention, and the typical behavior of the user. Moreover, it is an interesting question how a meaningful theoretical results for the IES can be achieved. In this paper, we intend to provide contributions to these questions. In particular we discuss new methods of how to conduct research in IES, analyze the user behavior in the selection process and study the feasibility of the self-adaptive step size adaptation within this context.

Our discussion adopts a representative problem for IES, namely the *re-design of RGB colors* by means of subjective evolution as suggested by Herdy [9]. The problem can be easily extended by using color patterns instead of a single color. Moreover, it can be easily explained to people participating in experimental studies, and thus can be readily used for collecting statistical data.

The structure of our paper is as follows: After a short introduction to ES (Section 2) we will compare different interaction modes for Interactive ES and discuss issues related to the convergence theory of the IES (Section 3). We continue with a discussion of self-adaptive features 4 in ES and discuss their role in interactive evolution strategies. Finally, in section 5, we report on first statistical studies of self-adaptive IES on color (pattern) redesign problems. The paper concludes (Section 7) with a summary of first results and an outline of some open questions for future research, some of which will be motivated by the results presented in this study.

2 Evolution Strategies

Next we will describe the (μ, κ, λ)-ES, a modern instantiation of the ES. The main loop of the (μ, κ, λ)-ES reads as follows: The algorithm starts with the initialization of a population (multi-set) P_0 of μ individuals (objective function vectors + mutation parameters). The initialization can be done uniformly distributed within the parameter space. P_0 forms a starting populations, and within subsequent iterations a series of populations $(P_t)_{i=1,2,...}$ is generated by means of a stochastic procedure: In a first step of this procedure λ random variations of individuals in P_t are generated by means of a variation operator, the details of which we will describe later. The new variants form the population Q_t (offspring population). Then, among all individuals in P_t and Q_t the μ best individuals that have not exceeded a maximal age of κ generations are selected by means of a selection criterion. In case of $\kappa = 1$ the strategy is termed (μ, λ)-ES, while in case of $\kappa = \infty$ we denote it with $(\mu + \lambda)$-ES.

The variation-selection process is meant to drive the populations into regions of better solutions as t increases. However, there is no criterion that can be used to determine whether the best region was found (except in cases with a pre-defined goal or bound on the objective space). Hence the process is usually terminated if the user decides to stop it, e.g. because of his/her time constraints or because of a long time stagnation of the best found value.

Next, let us describe the variation operator that are used to generate offsprings. Individuals within the ES (if applied for continuous optimization) consist of a vector of decision variables $\mathbf{x} = (x_1, \ldots, x_{n_x}) \in \mathbb{R}$ and a step-size vector $\mathbf{s} = (s_1, \ldots, s_{n_s}) \in \mathbb{R}^+$ that is used to scale the mutation distribution of an individual.

A mutation algorithm with a single step-size is described in algorithm 1. First the step-size of the parent individual is multiplied by a constant factor, the value of which can be 1, α or $1/\alpha$ depending on a random number. Then this step-size of the new individual is used to obtain the decision variables of the new individual. These are obtained by adding an offset to the corresponding value of the original individual. The value of this offset is determined by a standard normal distributed random number. The idea behind this mutation operator is that decision variable vectors that are generated with a favorable step-size are more likely to be part of the next generation, and thus also the information about the step size that was used to generate them is transferred to that generation. The process of mutative step-size adaptation was investigated in detail by Beyer et al. [5]. Due to his findings, simple adaptation rules like the 2-point or 3-point mutation for the step sizes serve well, whenever only a few iterations of the algorithm can be afforded. For a higher number of iterations, say $t_{max} \gg 100$, more sophisticated adaptation mechanisms should be considered for the parameters of the mutation. State of the art techniques include the individual step size adaptation by Schwefel [16] and the covariance matrix adaptation (CMA) by Hansen and Ostermeier [8]. Note, that in order to allow for a mutative step-size adaptation, a surplus of offspring individuals needs to be generated in each generation. The recommended ratio of $\mu/\lambda \simeq 1/7$ leads to a good average performance of the ES in many cases [16].

Algorithm 1 Generate λ offspring via 3-point mutation

```
 1: $Q = \emptyset$
 2: for $i \in 1 \dots \lambda$ do
 3:   choose $(\mathbf{x}, \mathbf{s})$ randomly out of $P_t$
 4:   $u \leftarrow$ uniform$(0, 1)$   // uniformly distributed random number between 0 and 1
 5:   $s_1' \leftarrow \begin{cases} s_1\alpha & \text{if } u < \frac{1}{3} \\ s_1/\alpha & \text{if } u > \frac{2}{3} \\ s_1 \text{ otherwise} \end{cases}$
 6:   for $j \in \{1, \dots, n_x\}$ do
 7:     $x_j' = x_j + s_1' \cdot$ normal$(0, 1)$
 8:     /* normal(0,1) generates standard normal distributed random number */
 9:   end for
10: end for
11: $Q = Q \cup \{(\mathbf{x}', \mathbf{s}')\}$
```

3 Interactive Evolution Strategies

3.1 Interaction Modes

There are many possibilities to integrate user interaction in the ES. In general, we can distinguish between reactive and proactive feedback. Reactive feedback is feedback requested by the algorithm, e.g.

- the user might be asked for evaluation (grading) of offspring individuals
- the user is asked for selecting individuals
- the user is asked for generating variants

In contrast to this, proactive feedback denotes an optional intervention by the user, e.g.:

- he/she might change the step-size parameter actively, e. g. after watching the search process stagnate
- he/she might insert a new individual into the population or actively change the variables of an individuals

In this paper we are more interesting in strategies with reactive feedback and the only proactive feedback will be given, when the user decides to stop the search process.

Three straightforward types to implement a selection procedure would be (1) to grade all individuals and let the algorithm select the best variations, (2) to sort all individuals, (3) to simply mark the μ best individual(s) as suggested in [9].

In order to compare these alternatives, let us introduce the *decision effort* as the number of possible choices the user has within one selection step. It seems reasonable to assume that this measure correlates positively with the felt effort by the user for conducting one selection. Following the procedure 1 would result in $(N_g)^\lambda$ possibilities, if N_g represents the number of possible grading levels. For the second alternative the user has to choose among $\lambda!$ possible sequences, while for procedure 3 the number of possibilities reduces to $\binom{\lambda}{\mu} = \frac{\lambda!}{(\lambda-\mu)!\mu!}$ possibilities. Accordingly, the decision effort for the latter would be the lowest (assuming $\lambda < N_g$). For $\mu = 1$ it even reduces to λ alternatives. Since, the user's time is usually limited in the following, we will focus on a strategy that minimizes the user's effort, namely alternative 3 and $\mu = 1$. Before discussing some experimental results with the latter strategy, let us first explore some further possibilities to conduct theoretical research on the behavior of interactive ES.

3.2 Bounding the Convergence Behavior

A problem when deriving a theory of interactive algorithms is that the behavior of the user is highly difficult to model. However, in the remainder of this section we explore possibilities to derive some meaningful theoretical results for the theory of IEA without explicitly modeling the user.

A model that is frequently used for the analysis of ES is that of a markov chain. A markov chain can be viewed as an autonomous stochastic automaton $(S, Pr\{s'|s\})$, where S denotes a state space, and Pr denotes a function that assigns a probability to each state transition from a state $s \in S$ to a subsequent state $s' \in S$, Accordingly, $\forall s \in S : \sum_{s' \in S} Pr\{s'|s\} = 1$. By setting $S = \mathbb{I}^\mu$, i.e. the space of possible parent populations, evolutionary algorithms on a finite search space can essentially be modeled as Markov chains. This allows to obtain results about the limit behavior and average behavior on some test problems (e.g. [3]).

It was suggested by Rudolph [15], to extend this model to a stochastic mealy automaton with deterministic output, whenever interactive evolutionary algorithms are to be modeled. Such an automaton can be denoted with $(S, X, Pr\{s'|s, x\})$, where X denotes a set of input symbols. Now, the probability function $Pr\{s'|x, s\}$ denotes the probability that under the condition that the current state is s and the user inputs the symbol x the next state will be s'.

An interesting observation is that given a stream of inputs, the behavior of this strategy is reduced to that of a Markov chain again, where by the input stream becomes part of the deterministic formulation of Pr. This allows us to separate the analysis of the interactive part from the, still stochastic, remaining part of the algorithm and, for instance, provides us with an means to analyze the best case behavior of a strategy for some target function f, by minimizing the expected distance $\mathrm{E}(\Delta_t|\mathbf{w}, f)$ to a desired target solution Δ_t over all possible user inputs $\mathbf{w} \in X^t$ for a pre-described number of iterations t:

$$\mathrm{E}(\Delta_t|\mathbf{w}^*, f) = \min_{\mathbf{w} \in X^t} \mathrm{E}(\Delta_t|\mathbf{w}, f). \tag{1}$$

Here X is for instance the space of all sequences of numbers between 1 and λ, in case of a simple selection scheme and $\mu = 1$. This convergence time of an 'ideal user' can be compared on test-cases with known result to the measured convergence behavior of an interactive ES in order to find out whether the user does the selections in an optimal way. If so, it is likely that the implementation of the algorithm marks the bottleneck of the convergence speed and it is not within the hands of the user to further improve the convergence speed. Otherwise, if the convergence speed of the 'ideal user' is indeed much faster than that of the interactive strategy, providing the user with selection aids or integrating some smart user monitoring strategy might help to further increase the algorithmic performance.

However, even for quite simple variant of interactive ES, the computation of an ideal user behavior might be a challenging task as in every step there are at least λ possibilities to choose the best individual, resulting in an exponential number of at least λ^t possibilities for input streams up to the t-th iteration. In some cases it might be possible to make the best choice in each iteration by means of theoretical considerations. It shall also be noted here, that it suffices to find a computer-based selection strategy that performs much better than the interactive ES to motivate the potential of further improvements of the user interaction. Later, we will give an example for such an analysis.

In summary, it seems that only in a few, very simple cases it will be possible to get meaningful results from a convergence theory of interactive ES and empirical results will likely play an important part in the dynamic convergence theory of these algorithms, even if we assume the 'ideal user'.

Another theoretical question that can be easily addressed would be to check if the strategy can converge globally. This would be the case, if the user can, in principle obtain any given starting point obtain any other state with a finite probability and a finite number of inputs, independent of the settings of the strategy parameters. However, such kind of considerations tell us nothing about the practically important convergence time, but it can well serve, to discard certain variants of interactive ES. For constant step sizes the result of probabilistic convergence in limit for regular target functions and ideal users is simply inherited from the theory of the non-interactive ES, provided that the best found solution is archived in case of a comma strategy. Moreover, the result can be extended to the self-adaptive case if the step-size is bounded below by a minimal positive step size [5].

4 Self-adaptation and Interaction

One of the questions addressed in this paper is, whether self-adaptive mechanisms of the ES works well for the interactive ES. With regards to this, there are some important differences between the standard ES and the interactive ES.

First of all, for the standard ES in continuous spaces, the precision of an optimum approximation can, in principle, get arbitrarily close. In applications of the interactive ES, the subjective nature of the objective function usually forbids an arbitrary close approximation of some solution. The reason for this is that in many cases the user will not be able to measure arbitrarily small differences in quality. For example, when comparing two colors, a human will perceive two colors as equal if their distance is below a *just noticeable difference*. The concept of JNDs is quite frequently discussed in the field of psycho-physics, a subbranch of cognitive psychology [2]. It is notable, that the JND depends on the intensity and complexity of the stimulus presented to the user. Moreover, it has been found that the lower the difference between two stimuli and the more complex the stimuli are, the more time it takes for the user to decide upon the similarity of two patterns. We will come back to this results, when we discuss the empirical results of our experiments.

Another difference between the standard ES and the ES with subjective selection criterion is that the user's attention level will decrease after a while. For a theory of attention we refer to Anderson [2] and Reason [13]. Hence, the number of experiments is usually very limited and very fast step-size adaptation mechanisms have to be found, and only a few parameters of the mutation distribution can be adapted.

Moreover, as discussed above, the amount of interaction should be minimized, e.g. by choosing a simple selection scheme. This might prevent the use of step-size adaptation strategies that demand for numerical values of the fitness function value. A performance measure would be based on the number of selections made by the user, rather than on the number of function evaluations.

5 A Color Redesign Test-Case

To study the effect of a human as fitness function and selection mechanism, a small experiment was constructed. An evolutionary algorithm was implemented in the form of a JAVA applet for the simple problem of finding the RGB values of a certain color or combination of colors (see figure 1). In this experiment the user selects one color or color pattern out of several alternatives that is closest to a given target color. This selection is then used as a parent for the next generation of alternatives that the user can choose from. When the user thinks the algorithm will not improve the results any more he/she can choose to stop the search by clicking the 'Done' button. All data collected in this applet is then send to a database and can be used for this research.

A one dimensional experiment was conducted using squares with only one color and a two dimensional experiment was done by having a left and a right color in the same square (see figure 1). Comparing these two experiments might give insight into the scalability of this type of experiment.

The whole faculty was asked to help with this experiment by running this applet in a web browser. About 200 runs were collected this way and the findings in the paper are based on these runs. This experiment was then also carried out using an 'ideal user' in the form of a computer program that would always select the color with the smallest Euclidian distance to the target color. Note, that

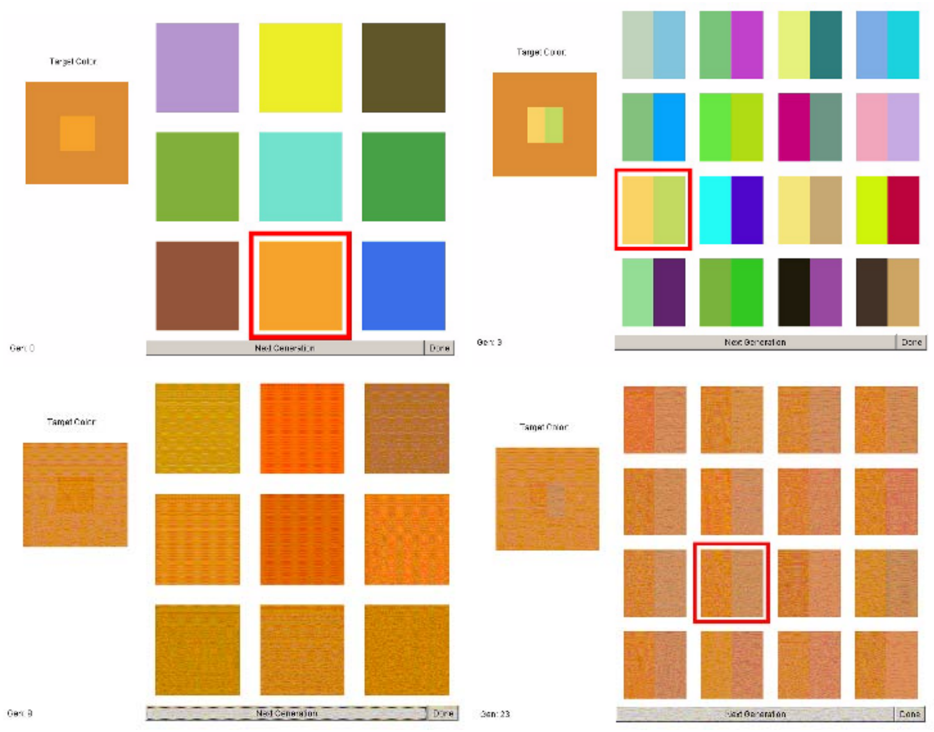


Fig. 1. Subjective selection dialogue with user: The upper figures show the initial color patterns (single color (left) and two-color test case (right)) and the lower figures show color patterns at a later stage of the evolution. The bigger box on the left hand side displays the respective target color, and in its center a small box is placed displaying the selected color. Once the user presses the NEXT bottom a selection gets confirmed and a new population will be generated and displayed. If the user is finally satisfied with the result he/she presses the Done button, in order to stop the process.

this strategy maximizes the convergence speed in case of an ES with constant step size. Having data on both a deterministic selection compared to a human selector, provides us with insights of whether the user selects individuals in a way that maximizes the convergence speed.

Two different algorithms were used. One with a fixed step size and one with a self adapting step size. Three different stepsizes were used in the fixed algorithm: 10, 20 and 40. Note that RGB values can range from 0 to 255, that makes the relative stepsizes approximately 0.039, 0.078 and 0.156. For the self adaptation the 3 point step-size mutation scheme by Rechenberg was employed with $\alpha = 1.3$ (cf. algorithm 1).

When the applet is started an algorithm is randomly selected for that run. The user will only know what algorithm was selected after the experiment so that the user will not be influenced by that knowledge. The random generation was done in such a way that 50% of the runs were done with the Rechenberg algorithm and 16.667% of the runs for everyone of the three fixed step-size algorithm.

The target color was fixed in all the experiments to make them comparable and was chosen to be $[R = 220, G = 140, B = 50]$ so that every component of the color was neither close to the variable bounds 0 and 255 nor equal or close to any other component. Our choice happened to be a brownish version of orange.

6 Results

First results we collected with our JAVA applet, by the help of many users who participated in the experiment, are displayed in figures 2 and 3. Next, we will discuss these results one by one.

Figure 2 shows the average function values for the different strategies. Note that not all runs had the same length. Some people put more effort into finding a better result than others. This is all part of the effect of using humans instead of a computer. Conclusions based on this data should take this into account. It is also notable that users took on average 5 seconds for a selection and hardly proceeded more than 40 iterations.

The plot in figure 2 shows that self adaptation seems to outperform all the other algorithms, with human selection as well as with the computer selection. The fixed algorithm with step-size 20 seems to converge a lot faster in the beginning though and stagnate after 10 iterations. This effect is also very plain with the computer selection. There the fixed is quicker until generation 10 and then self adaptation closes the gap and overtakes. The computer selection was also run on the three different fixed step sized, but step size 10 was by far the best in both the one dimensional and the two dimensional case. We note here, that it is hard to figure out a-priori what is a good step size for an experiment with

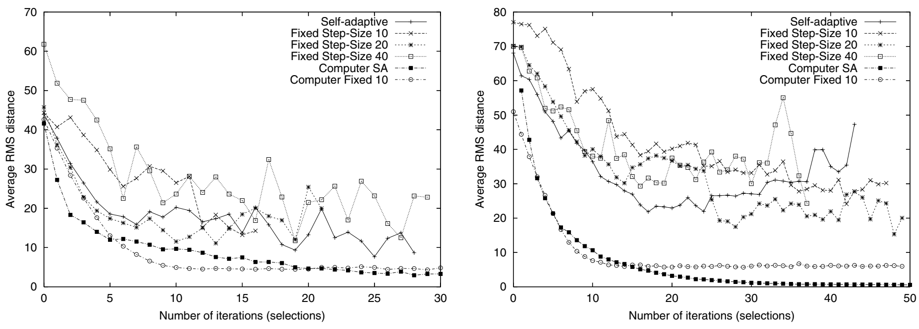
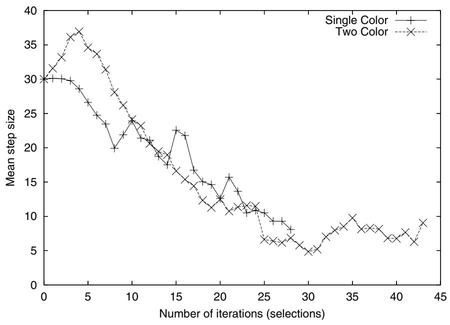


Fig. 2. The convergence behavior of different ES obtained in the online experiment. The upper figure displays results for a single RGB color, and the lower figure results for two different RGB colors. The number of iterations depends on the number of iterations the at least two of the users spend until terminating the experiment.



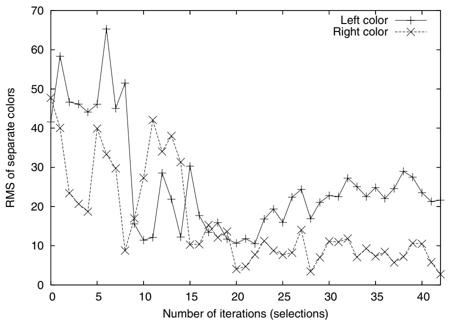


Fig. 3. The left plot shows step-size adaptation in interactive ES for single color and two-color example. The right plot displays the history of a typical run of the two color problem. The distance of the two colors to the target color is displayed over the number of iterations. Note the divergence of the left color after 20 iterations.

subjective evolution, and thus a self-adaptive step size will be always a favorable solution if no additional problem knowledge is given.

The two dimensional plot is a bit different though. Here a fixed step size of 20 seems to be the better choice. Self adaptation performs well in the beginning best here but after 20 iterations the error seems to go up again. This is a unexpected effect that only seems to occur with the self adaptation in the two dimensional case if a human selection is used. In the case of the computer selection this effect is totally absent and self adaptation outperforms all the fixed step-sizes.

In an effort to explain this effect figure 3 (right) shows the error of both colors separately in a typical run of self adaptation using human selection. Note, how both errors seem to converge nicely up until generation 20. From that point onward only the right color seems to converge to an optimum whereas the left color is actually moving away from the optimum. This suggests that this shows how a human might try to use a certain strategy to optimize two dimensions. In this case it seems the user tried to optimize the right color first and worry about the left color later. Although this seems a feasible strategy, the self adaptive algorithm has some problems with that. The total error goes up, the step-size is stagnating and even increasing a bit (as figure 3 (left) shows).

What figure 3 also shows is that the step-sizes are decreasing at the same rate in the one dimensional problem as they do in the two dimensional problem. It seems though that in the two dimensional problem the step-size starts of a bit higher.

The fact that the computer selection outperforms the human selection is not very surprising, as the selection could be viewed as a noisy selection method. However, it is quite surprising how much this noise influences the algorithm. A conclusion from this is that there might be still a great potential for improvements of the performance, that might be reached by assisting the user and diminishing the noise. Moreover, when analyzing the noise, there seems to be more to it than just adding noise to a fitness function. The results suggest humans use strategies that are based on some outside knowledge or even feelings that influences self adaptation in an unfavorable way. Moreover, cognitive restrictions of humans with regard to attention and just noticeable differences will have to be taken into account when modeling this noise.

7 Conclusion

This paper contributes to the analysis of Interactive Evolution Strategies (IES) with subjective selection criteria. The approach has been related to the context of human algorithm interaction. Differences between interactive evolutions to non-interactive ones were pointed out, including a discussion of different ways to analyze these methods. In particular, we compared different forms of interaction, e.g. by means of the decision effort criterion, and suggested concepts to obtain bounds on the performance of interactive EA. Here we introduced the concept of an 'ideal user' that can be used to estimate the potential for improvements in the interactive part of the algorithm.

In the empirical part of the paper we added new results to the experimental work started by Herdy [9] on a color re-design test case. The experiment was extended by a two-color redesign example. A JAVA applet implementing the IES on the color test cases was developed and used in an internet survey to obtain a significant number of results from different users.

The results clearly indicate the benefit of the step-size adaptation. Strategies that work with step-size adaptation turned out to be more robust and diminish the risk to choose a completely wrong step size. It is notable within this context, that the employed 3 point step-size adaptation proved to be beneficial for a very small number of less than forty generations.

By comparing to the results obtained with an 'ideal' user's selection scheme, we could show that the users did hardly select individuals in a way that maximizes the convergence speed. This adds evidence to the fact that the noisy nature of the user-based selection is harmful to the algorithm's behavior. However, this result can also be interpreted in a positive way, as it shows that there is still much room for improvements for the user interaction. For instance decision aids, noise reduction strategies, or smart user monitoring strategies might help to further increase the performance of the IES.

We also note, that the kind of selection errors made by the users can hardly be modeled by standard noise models like constant gaussian distributed offsets to the objective function values. The noise function seems to be time dependent and dependent on the distance to the target values.

For more complex targets another aspect has to be taken into account when modeling the user: An insight we got from observations for more complex target definitions (two color example) was that the user starts to use some strategy, e.g. to first optimize the first and then the second color. Such kind of user behavior has rarely been addressed in the context of interactive evolutionary algorithms and deserves further attention.

Acknowledgements. Michael Emmerich and Thomas Bäck gratefully acknowledges financial support by the HEFBOOM NWO Project. Also we acknowledge the support of the people from LIACS who participated in the experiments.

References

1. P.J. Angeline (1996): Evolving fractal movies, 1st annual conference on generic programming (Stanford, CA, USA), pp. 503-511
2. Anderson J.R. (2004): Cognitive Psychology and its implications, Worth Publishers, UK
3. Bäck, T. (1996). Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York, 1996.
4. Banzhaf, W.: Interactive evolution. In: Handbook of Evolutionary Computation (T. Bäck,, D. Fogel, and Z. Michalewicz, ch C2.10, pp. 1-5, Oxford University Press, 1997
5. Beyer, H.-G. (2001). The Theory of Evolution Strategies. Springer, Berlin, 2001
6. Dix, A., Finlay, J., Abouwd, G.D., Beale, R. (2003). Human Computer Interaction (3rd ed.). Pearson Education, 2003.
7. B. Filipic, D. Juricic (2003): An interactive genetic algorithm for controller parameter optimization, Intl. Conf. on Artificial Neural Nets and Genetic Algorithms, Innsbruck, Austria, pp 458–462
8. Hansen, N. and Ostermeier, A. (2001). Completley Derandomized Selfadaptation in Evolution Strategies, Evolutionary Computation, 9(2):159-195, 2001.
9. M. Herdy (1996): Evolution strategies with subjective selection, PPSN IV, Berlin, Germany, 1996
10. M. Herdy (1997) Evolutionary optimization based on subjective selection - evolving blends of coffee, In: 5th european congress on Intelligent Techniques and Soft Computing EUFIT'97, pp.640-644
11. Horowitz (1994) Generating rhythms with genetic algorithms, in Int. Computer Music Conference, (ICMC'94) (Aarhus, Denmark), pp. 142– 143, 1994
12. I.C. Parmee, C.R. Bonham: Cluster oriented genetic algorithms to support designer/evolutionary computation, Proc. of CEC'99, Washington D.C., USA, 546-55
13. Reason J. (1990): Human Error, Cambridge University Press, Cambridge UK, 1990
14. Rechenberg, I. (1994). Evolutionsstrategie '94. Frommann-holzboog, Stuttgart, 1994.
15. Rudolph, G.: On Interactive Evolutionary Algorithms and Stochastic Mealy Automata. PPSN 1996: 218-226.
16. Schwefel, H.-P. (1995), Evolution and Optimum Seeking, Wiley, NY
17. Takagi H.(2001) "Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation", Proceedings of the IEEE, vol.89, no.9, pp.1275-1296, 2001.

An Experimental Comparative Study for Interactive Evolutionary Computation Problems

Yago Sáez[1], Pedro Isasi[1], Javier Segovia[2], and Asunción Mochón[3]

[1] Universidad CARLOS III de Madrid, Leganés 28911, Spain
yago.saez@uc3m.es
http://et.evannai.inf.uc3m.es/personal/ysaez/
[2] Universidad Politécnica de Madrid, Facultad de Informática, Campus Montegancedo, Boadilla del Monte 28040, Spain
[3] Universidad Nacional de Educación a Distancia, Departamento de Economía, Aplicada, Madrid 28040, Spain

Abstract. This paper presents an objective experimental comparative study between four algorithms: the Genetic Algorithm, the Fitness Prediction Genetic Algorithm, the Population Based Incremental Learning algorithm and the purposed method based on the Chromosome Appearance Probability Matrix. The comparative is done with a non subjective evaluation function. The main objective is to validate the efficiency of several methods in Interactive Evolutionary Computation environments. The most important constraint of working within those environments is the user interaction, which affects the results adding time restrictions for the experimentation stage and subjectivity to the validation. The experiments done in this paper replace user interaction with several approaches avoiding user limitations. So far, the results show the efficiency of the purposed algorithm in terms of quality of solutions and convergence speed, two known keys to decrease the user fatigue.

1 Introduction

Evolutionary Computation (EC) encompasses computational models which follow a biological evolution metaphor. The success of these techniques is based on the maintenance of genetic diversity, for which it is necessary to work with large populations. The population size that guarantees an optimal solution in a short time has been a topic of intense research [2], [3]. Large populations generally converge to better solutions, but they require more computational cost and memory requirements. Goldberg et al. [4] developed the first population-sizing equation based on the variance of fitness. They further enhanced the equation which allows accurate statistical decision making among competing building blocks (BBs) [2]. Extending the decision model presented in [2], Harik et al. [3] tried to determine an adequate population size which guarantees a solution with the desired quality. To show the real importance of the population size in Evolutionary Algorithms (EAs) He and Yao [5] showed that the introduction of a non random population decreases convergence time. However, it is not always possible to deal with such large populations, for example, when the adequacy values must be estimated by a human being (Interactive Evolutionary Computation, IEC). Those environments require methods capable to perform well with a short number of individuals (micropopulations).

IEC techniques can be applied to different fields, such as digital treatment of images [6], composition of musical works [7], automotive design [8], automatic design of figures [9], general artistic design [10], [11], for example, the design of sculptures [12], or generation of gestures in 3D figures, aimed at virtual worlds, pictures or games, [13]. A more detailed sort of examples can be found in a complete survey about IEC [14].

The main problem in most of the above-mentioned references is that the selection criteria is based on a merely artistic and personal point of view. In these kind of applications the problems become more complex since the criteria used is subjective, (it is based on human perception and changes according to the user's opinions or personal preferences). To avoid this problem a typical IEC design problem with a non-subjective evaluation function is presented. This environment allows to run a large number of experiments and to analyze the behaviour of the four selected algorithms under the same conditions.

The remainder of the article is organized in the following way: a brief description of the algorithms used to compare the performance of our proposal is presented in section 2. Section 3 reports the description of the problem environment and the results of the comparative tests. Finally, the conclusions are drawn in section 4.

2 Description of Algorithms

IEC algorithms are difficult to test because they require the evaluation of the user in their experiments. In addition, if the designer wants to compare the new algorithms with existing ones, even more experimentation with humans is required. The proposed algorithm (Chromosome Appearance Probability Matrix, CAPM) is compared in every domain with the classical Genetic Algorithm (Simple Genetic Algorithm, SGA) with the Fitness Predictive Genetic Algorithm (FPGA) and a probabilistic algorithm called Population Based Incremental Learning (PBIL). The SGA is always a good point of reference for the comparison and the FPGA is one of the latest proposals for IEC problems [15]. The probabilistic approach gives another interesting point of view to compare with, because it is the starting point for the Estimation of Distribution Algorithms (EDA), [16] and [17]. Besides, the proposed method was partially inspired by the EDAs.

The representation chosen for all the chromosomes is the same in all the contrasted algorithms. Each individual is made up of various chromosomes which are in turn, made up of a vector of integers. In the following section the selected algorithms, except the SGA, will be briefly explained.

2.1 Fitness Predictive Genetic Algorithm

Since the user is present during the evaluation and the selection process in IEC techniques it is necessary to work with micropopulations which are easily evaluated. Nevertheless A small number or a reduced population affects negatively the productivity of the GAs and, as a result, improvements over the conventional GA are proposed. These improvements depend on the size of the population and the fitness prediction of all the individuals which are not evaluated by the user, [15], [27]. Thus it deals with an algorithm of M individuals and only a subset of N are shown to the user. Then the user makes its personal evaluation. Finally, the fitness of the rest (M-N) is obtained according to the differences encountered with the selection made by the user in the previous iteration. There are several types of approaches in order to estimate the fitness of the rest, but only the most simple has been developed for this paper: the Euclidean distance approach. The FPGA follows the next steps:

Step 1 - Initiate the population with N number of possible random solutions: during this process N number of individuals are randomly initiated from the population. The value of N depends on the quantity of individuals which want to be shown to the user per iteration and it is also linked to the type of problem which wants to be solved. The experiments done have been based on N=10.

Step 2 - Evaluate N candidates: this process is responsible for the evaluation of N visible candidates just as the user would do. As a result, and if decoding was necessary, each of the genotypes are converted to corresponding phenotypes. With the fitness value or adaptation of each individual of the whole N assigned, the determined evaluation function for each problem is applied. At this point, all other individuals of the population (M-N) remain to be evaluated. In the experimental domain populations of 100 individuals have been used, out of which the user evaluates the best 10, M=100 and N=10.

Step 3 - Forecast fitness values of candidates M-N: this process is used if the condition which guarantees the number of iterations is greater than 2. Alternatively, it is used when an initial reference evaluation has been made which allows the forecasting of the fitness values of the rest of the population (M-N). The necessary modifications to adapt the algorithm to the selection method have been done carefully and it corresponds to the proposals regarding to the algorithm put forward by Hsu, [15]. Predictive fitness is obtained by the calculation of Euclidean distance between the referenced chromosomes (those selected by the user and those of the individuals not evaluated by the user). FPGAs are effective in typical IEC problems in which the parameters are coded in such a way that the Euclidean distance gives an idea of how close or far an individual is from the one selected by the user. As the FPGA is not designed to be applied to numerical optimization problems and is not foreseen as being used with the proposed coding, it is possible that due to their genotype differences, two very close variables (as far as the phenotype is concerned) could produce great distances.

Step 4 - Select two parents: as with the SGA, the selection operator implemented for the experimentation is based on the selection of the two best individuals according to their adaptation or fitness value. However, the selection can only be done with the best 'N' individuals of the population. This previously mentioned limitation is typical of FPGAs and IEC techniques in which the user evaluates. Therefore, the selection forces the evaluation to a subset of the population 'N'. During each iteration, the selections made by the user are stored in the ps_1 and ps_2 variables with the intention of doing other calculations which will obtain the predictive fitness of the following iterations.

Step 5 - Cross pairs with parents: this process is exactly the same as the SGA. Once the best two individuals of fitness value are selected the cross operator is applied in order to obtain the new generation of individuals. The propagation strategy is elitist and, as a result, the selected parents go directly to the following generation. The remainder are generated from the equally probable crossing of the parents.

Step 6 - Mutate the obtained descendants: as with the SGA, this process is responsible for mutating, with a certain probability, ($P_{mutation}$) several genes of the generated individuals. The aim is to guarantee the appearance of new characteristics in the following populations and to maintain enough genetic diversity.

Step 7 - Repeat until the final condition: like the SGA, the stop condition of the algorithm is imposed by a maximum limit of iterations.

2.2 Population Based Incremental Learning (PBIL)

The basis for introducing learning in GA's were established with the Population Based Incremental Learning (PBIL) algorithm proposed by Baluja and Caruana (1995), [23]. This algorithm means another optimization approach different than the GAs, obtaining excellent results in certain problems, like [18], [19] and [20]. Besides, it has been the starting point for EDAs, [16] and [17], and a good example for the proposed method CAPM.

The PBIL algorithm, [23] is partially based on the Equilibrium Genetic Algorithm (EGA), [21], but with some improvements made. The authors present it like a mixture between an EC algorithm and a hillclimbing approach, [18].

The main concept of the PBIL algorithm is the substitution of the genetic population with a set of statistics representing the information about the individuals. Therefore, the selection and crossover operators are not needed anymore, and a probability distribution process is the responsible for changing the populations each iteration.

The success of this statistic approach opened a wide research field with lot of works, [24], [16], [17], [26], etc..

Step 1 - Initialize probability matrix: this procedure is responsible for initializing randomly all the individuals, also called solution vectors. For this task the probability matrix is firstly initialized with uniform distribution, equation 1, and all the alphabet elements will have the same likelihood of appearing.

$$P_{i=[0..(k-1)],j=[0..(L-1)]} = \frac{1.0}{k} \tag{1}$$

where k represents the alphabet size and L the chromosome size.

Step 2 - Evaluate and generate N vectors of solution: this procedure deals with the evaluation of the solution vectors. The solution vectors are generated after the first iteration through the sampling of the probability matrix. Once the best solutions are selected, they are used to update the information stored in the probability matrix.

Step 3 - Update the probability matrix: this step uses the following updating rule:

$$P_i(X = j) = (P_i(X = j) \times (1.0 - \alpha)) + (\alpha \times M_{i,j}) \tag{2}$$

where $P_i(X = j)$ is the probability of generate any of the j values which belong to the alphabet in the i^{th} position of the chromosome. α is the learning factor, and $M_{i,j}$ the probability matrix, column i, row j.

The learning factor of the algorithm (α) can differ depending on the problem. Besides, it affects the final results, [23]. Smaller learning rates imply wider searches, and higher values mean deeper search processes. However, its value should not be too high, because as the learning factor increases, the dependency between the solution and the initial population is higher.

Step 4 - Mutation of the probability matrix: the mutation operator plays an important role during the search process in order to guarantee the convergence avoiding local optimums and maintaining the diversity through the iterations.

The mutation operator in PBIL algorithms can be done at two levels: solution vector or probability matrix. Both of them are useful for maintaining the genetic diversity, but after several experiments made in different works, the probability matrix mutation appears to be slightly better, [22] or [23].

Step 5 - Repeat until the final condition: like the SGA and the FPGA the stop criterion is imposed by a maximum limit of iterations which depend on the problem to solve.

2.3 Chromosome Appearance Probability Matrix

In PBIL algorithms, the recombination operator is replaced by a vector of independent probabilities of each variable, and sampling this vector implies the study of the selections made by the algorithm till that moment. This concept, applied to IEC can be done in order to speed up the evolution in regards to the user needs. This was the key motivation for developing this new method based on the Chromosome Appearance Probability Matrix.

The steps of the proposed algorithm are explained in detail in [28] and [1], however this method introduces the following new features which differ from a canonical genetic algorithm.

Probability matrix for guiding mutation: when the user selects an element of the population, his or her selection is usually based on the collective combination of features in each element of an individual. For example, if the user is searching for tables he will appreciate the combination of several characteristics such as the color of legs of the table, the number, the shape of the surface, etc.. Therefore the information about the whole chromosome should be kept. To do this, a multidimensional array, with the same number of dimensions as genes, has been included. The bounds of the dimensions are determined by the number of alleles of the different genes.

The probability array 'M' is initialized by $M(gene_1, gene_2, gene_3, gene_m) = 1/T$ where 'm' is the number of genes, and gene$_i$ could have values in $[allele_1^i$, $allele_2^i \ldots allele_{n_i}^i]$, and n_i the number of alleles of gene 'i'. The total possible combinations of chromosomes 'T' is calculated by multiplying the maximum sizes of each gene ($T = \prod_{i=1}^{m} n_i$).

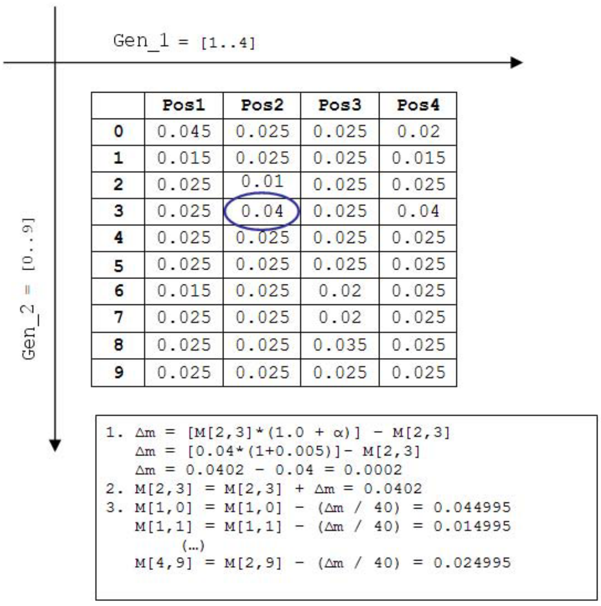
This array shows the probability of being chosen that each possible combination of alleles have. Each iteration implies a selection of one or two individuals, and its chromosomes represent a position in the above array. After the selection, the corresponding position in the array is updated by a factor of α with the increment factor of the update rule, Δ_M. This Δ_M is calculated by the following equation:

$$\Delta_M = [M_{gene_s^1, \cdots, gene_s^n} \times (1.0 + \alpha))] - M_{gene_s^1, \cdots, gene_s^n} \qquad (3)$$

The example in figure 1 shows how the update rule works for 1 chromosome with 2 different genes, gen_1 with 4 alleles {pos1,..,pos4}, and gen_2 with 10, {0..9}. It can be clearly seen how the probability matrix 'M' is updated with $\alpha = 0.005$ and how it affects to the rest cells.

The update operations take care that the sum of all the elements of the array will be 1. This array is very useful to keep information about the selection frequency of a determined chromosome, and therefore, to help the mutation process to evolve towards the preferences of the user.

Oriented mutation operator: the mutation operator is responsible for the mutation of the individuals. Once a gene has been selected for mutation, a specific chromosome is taken as the base of the mutation process (reference chromosome). This chromosome is selected from the whole possible chromosomes following a uniform distribution fixed by the probability array. The higher the value of a chromosome's likelihood array, the better the probability of being chosen. In the mutation process, a position in the chromosome to be mutated is randomly selected. Then, the gene in this position is substituted by a gene from the reference chromosome in that same position. Thus, the mutation operator is the result of a function from the chromosome to be mutated and the reference chromosome:



	Pos1	Pos2	Pos3	Pos4
0	0.045	0.025	0.025	0.02
1	0.015	0.025	0.025	0.015
2	0.025	0.01	0.025	0.025
3	0.025	0.04	0.025	0.04
4	0.025	0.025	0.025	0.025
5	0.025	0.025	0.025	0.025
6	0.015	0.025	0.02	0.025
7	0.025	0.025	0.02	0.025
8	0.025	0.025	0.035	0.025
9	0.025	0.025	0.025	0.025

```
1. Δm = [M[2,3]*(1.0 + α)] - M[2,3]
   Δm = [0.04*(1+0.005)]- M[2,3]
   Δm = 0.0402 - 0.04 = 0.0002
2. M[2,3] = M[2,3] + Δm = 0.0402
3. M[1,0] = M[1,0] - (Δm / 40) = 0.044995
   M[1,1] = M[1,1] - (Δm / 40) = 0.014995
       (…)
   M[4,9] = M[2,9] - (Δm / 40) = 0.024995
```

Fig. 1. Update rule example for CAPM algorithm, $\alpha = 0.005$

$$newChrom' = mutateOneGen(OldChrom, referenceChrom) \tag{4}$$

This approach has the ability to broadcast the preferences of the user towards all the chromosomes of the individuals.

The inclusion of a clone remover operator: the clone remover operator is responsible for mutating all those individuals which have exactly the same genetic structure as the other individuals in the same population.

Replacement of all the population but parents with the new individuals: the proposed strategy is elitist, however, as the user is not interested in evaluating the same individuals between iterations, the algorithm mutates the parents for the next generation, making them slightly different.

3 Experimental Tests

3.1 Trademark Finder

The problem arises from an idea proposed in [25] and it is explained more in detail in [1]. The challenge of the trademark finder is to help the user in the task of finding a specific logo which is new, different, or eye-catching, for a product, or a company. For this purpose, like in brainstorming process, the system offers different types of words, applying different colors, backgrounds, and styles. This is only a first version for making the experiments, but in future developments figures and letter positions should change.

The algorithm starts randomly with six words (population), each one made of five letters, with random colors and styles. The user must select in each iteration the best two alternatives for him. Thus, each word is an individual of the population, and each letter is a chromosome with four genes representing color, font type, size and background, respectively. The search space is $3,2 \times 10^{21}$.

3.2 Experimental Framework

Numerous experiments (≥ 100.000) are required to validate empirically the efficiency of the methods. It must be taken into account that the problem should be run repeatedly for all the possible parameters and algorithms. Besides the fitness function must be the same for all the experiments in order to do an homogeneous comparative of the results. This is impossible to achieve with users (individual subjective preferences). Therefore, it is necessary to develop a simulator which replaces the human interaction acting like a virtual user equal for all the experiments, no matter the algorithm tested.

Evaluation based on ranking of preferences: for some type of products and business the preferences of corporate colors and fonts can be clear at the beginning of the search. Perhaps all letters in red are preferred, but the type of red can be selected by the user. For this reason, in the first test, an automatic evaluation which simulates the user behavior was designed with a ranking of preferences evaluation function.

However, the problem with this first test is that the automatic evaluation function is a non real world based function. To make the problem more complex it has been decided to include more realistic conditions which means non linear evaluation function and fluctuation in the preferences of the virtual user.

Evaluation based on fluctuating decisions rules sets: like humans do, it is possible that the user could find more than one type of solution interesting. Furthermore, after the study of the experiments conducted with humans [25] it was realized that very often the user does not have a clear idea of his own preferences. Also, often happened that his preferences or criteria changed as new designs were proposed. To simulate those doubts inherent to human behaviours, two sets of predefined evaluation rules which confront different preferences were included. Those rules randomly change with a probability of 70% using set 1, and 30% using set 2 (each iteration). These rules are applied independently to each chromosome (character) to obtain an objective measure.

In order to make even harder the search two conditions were included. These conditions forced the solution to search words with the same background for first and last letter and different for the remainder.

3.3 Comparative Study

This section presents a comparative study of some simulations in order to show the behavior of the algorithms explained in section 2. In the experiments developed the final condition is reached when the fitness of one element is 100 points (optimal solution), or when the iterations surpass 50. A valid measure is consider when the fitness is greater than 90.

At this point, after making 10.000 experiments per algorithm, running them with different parameters, see table 1, and both types of automatic evaluation function, the following results were achieved (best parameters found for each algorithm):

For the evaluation function based on the ranking the preferences only the CAPM method reaches the optimal solution in all experiments done (iteration 32 in average), and neither of the others even reached a valid solution.

When comparing the results, there is a significant difference between the algorithms: CAPM reaches much faster solutions in terms of iterations and better quality solutions. As can been seen clearly in tables 1, 2 and also in figure 2.

Besides, the PBIL shows clearly a low performance when working with micropopulations, due to the genetic diversity during the first 50 iterations is too low. Experiments with mutation likelihood of 45% have been run but this high mutation rate does not help the algorithm and makes the search completely random.

The FPGA has a good performance, but does not always beat the SGA results. The main reason is that the Euclidean distance equation used for the fitness prediction is not useful when predicting ranking evaluations or fluctuating rule sets. At this point future developments to try FPGA with prediction functions based on Neural Networks or Support Vector Machines are suggested.

Looking at the results for the fluctuating decisions evaluation model, the user must take at least 35 iterations in average to obtain a valid solution, and never reaches an optimal solution. However, the results are no so bad taking into account that the evaluation is using a non-linear function which changes of evaluation rules randomly (with 30% of probabilities of changing each iteration).

Table 1. Parameters used for experiments

Algorithm	Population Size	Mutation Prob.	Learning factor (α)
SGA	10	[5% . . . 45%]	N/A
FPGA	10(100)	[5% . . . 45%]	N/A
PBIL	10	[5% . . . 45%]	$[0,01 \ldots 0,1]$
CAPM	10	[5% . . . 45%]	$[0,001 \ldots 0,01]$

Table 2. Results with ranking of preferences

Iteration	SGA_10	$FPGA\_100$	$PBIL\_10$	$CAPM\_10$
10	50,00	51,00	30,00	62,00
20	65,00	65,00	35,00	85,00
30	74,00	74,00	38,00	**98,00**
40	80,00	80,00	43,00	**100(32)**
50	85,00	84,50	49,00	**100**

Table 3. Results with fluctuating decisions rule sets

Iteration	SGA_10	$FPGA\_100$	$PBIL\_10$	$CAPM\_10$
10	51,25	52,00	35,50	58,00
20	63,50	61,50	37,00	71,50
30	70,00	70,00	39,50	84,50
40	76,00	74,50	40,50	**91,50**
50	80,00	77,99	44,50	**96,00**

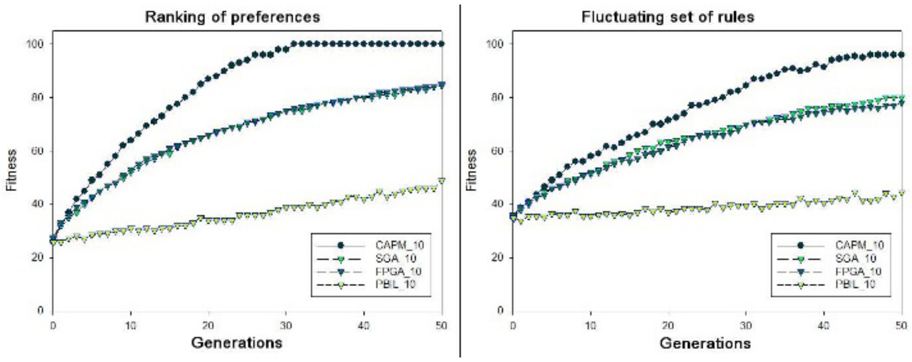


Fig. 2. Average fitness per iteration in both evaluation functions

It applies opposite evaluation preferences for each set of rules, and finally finds valid solutions after 35 iterations in average. The evolve process is very complex and is unpredictable what is really going to happen with the user. It is not usual to see the user changing completely of preferences each iteration (with 30% of probabilities), but it was decided to experiment with the worst case. In fact, in this evaluation mode all the algorithms tested except CAPM are unable to find even a valid solution (fitness ≥ 90).

4 Conclusion

After the study and development of different algorithms and their formal comparative test for solving an specific problem of IEC, the following conclusions have been drawn:

1. To test formally algorithms in IEC frameworks it is necessary to run a large number of experiments. This task can only be achieved when working with automatic evaluation functions. These functions must simulate human characteristics, which are not always predictable. For this paper the simulator changes its opinions or preferences randomly, with a probability of 30%. Also, it has been decided to make the search based on a non linear fitness function.

2. The analysis of the results of the SGA, shows that, although it is considered sufficient so far, it is not good enough for IEC, because it never reaches a valid solution in the experiments made. Besides, the proposed method has been compared successfully with an algorithm specifically developed for solving IEC problems (FPGA) and with a probabilistic algorithm (PBIL).
3. The proposed method has the capability of learning by the study of the selections made by the user. This alternative improves the results given by the SGA, FPGA and PBIL in terms of the quality of the solutions and the number of iterations in finding valid/optimal solutions.

   Finally, as the micropopulations affects negatively to all algorithms tested, becoming the main reason to decrease their performance, to avoid this problem a proposal is made to increase the selection pressure with the probability matrix. However, other proposals based on predictive fitness must be studied too.

Acknowledgment. This article has been financed by the Spanish founded research MCyT project OPLINK, Ref: TIN2006-08818-C04-02.

References

1. Sáez Y., Isasi P., Segovia J., J.C. Hern'andez "Reference chromosome to overcome user fatigue in IEC," *New Generation Computing, vol. 23, number 2, Ohmsha - Springer, pp. 129–142* (2005).
2. Goldberg D. E., Deb K., and Clark J. H., "Genetic algorithms, noise, and the sizing of populations," *Complex Syst., vol. 6, no. 4*, pp. 333-362. (1992).
3. Harik G., Cantú-Paz E., Goldberg D. E., and Miller B. L., "The Gambler's ruin problem, genetic algorithms, and the sizing of populations," *Transactions on Evolutionary Computation, vol. 7*, pp. 231–253, (1999).
4. Goldberg D.E., and Rundnick M. "Genetic algorithms and the variance of fitness", *Complex Systems, vol. 5, no. 3*, pp. 265–278, (1991).
5. J. He and X. Yao, "From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms," *IEEE Transactions on Evolutionary Computation, vol. 6*, pp. 495–511, Oct. (2002).
6. Sims K., "Artificial Evolution for Computer Graphics," *Comp. Graphics, Vol. 25, 4*, pp. 319–328. (1991).
7. Moore, J.H. "GAMusic: Genetic algorithm to evolve musical melodies." *Windows 3.1 Software available in: http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/genetic/ga/systems/gamusic/0.html.* (1994).
8. Graf J., Banzhaf W. "Interactive Evolutionary Algorithms in Design. Procs of Artificial Neural Nets and Genetic Algorithms, *Ales, France*, pp. 227–230, (1995).
9. F.J. Vico, F.J. Veredas, J.M. Bravo, J. Almaraz "Automatic design sinthesis with artificial intelligence techniques." *Artificial Intelligence in Engineering 13*, pp. 251–256, (1999).
10. Santos A., Dorado J., Romero J., Arcay B., Rodríguez J. "Artistic Evolutionary Computer Systems," *Proc. of the GECCO Workshop, Las Vegas.* (2000).
11. Unemi T. "SBART 2.4: an IEC Tool for Creating 2D images, movies and collage, Proc. of the Genetic and Evolutionary Computation" *Conference Program, Las Vegas.* (2000).

12. Rowland D. "Evolutionary Co-operative Design Methodology: The genetic sculpture park." *Proc. of the GECCO Workshop, Las Vegas.* (2000).
13. Bentley P., "From Coffee Tables to Hospitals: Generic Evolutionary Design," *Evolutionary design by computers, Morgan-Kauffman*, pp. 405–423. (1999).
14. Takagi H. "Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation," *Proc. of the IEEE, vol 89, number 9*, pp. 1275–1296 (2001).
15. Hsu F.-C. and Chen J.-S., "A study on multi criteria decision making model: Interactive genetic algorithms approach," *IEEE Int. Conf. on System, Man, and Cybernetics (SMC99)*, pp. 634–639 (1999).
16. Larrañaga P. and Lozano J. A., "Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation." *Boston, MA: Kluwer*, (2001).
17. Larrañaga P. and Lozano J. A., Bengoetxea E. "Estimation of distribution algorithms based on multivariate normal and Gaussian networks" *KZZA-IK-1-01*, (2001).
18. Baluja S., "An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics" *CMU-CS-95-193*, (1995).
19. Sebag M. and Ducoulombier A., "Extending Population-Based Incremental Learning to Continuous Search Spaces" *Lecture Notes in Computer Science", vol 1498*, pp. 418–426 (1998).,
20. Monmarché N. and Ramat E. and Dromel G. and Slimane M. and Venturini G., "On the similarities between AS, BSC and PBIL: toward the birth of a new metaheuristic" *citeseer.ist.psu.edu/557467.html*, (1999).
21. Juels A. and Baluja S. and Sinclair A., "The Equilibrium Genetic Algorithm and the Role of Crossover", *citeseer.ist.psu.edu/juels93equilibrium.html*, (1993).
22. Juels A. and Wattenberg M., "Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms", *Department of Computer Science, University of California at Berkeley*, (1995).
23. Baluja S., Pomerleau D., Jochem T.", "Towards Automated Artificial Evolution for Computer-generated Images" *Connection Science* 325–354, (1994).
24. Gonzalez C., Lozano J., Larranarraga P. "Analyzing the PBIL algorithm by means of discrete dynamical systems.", *Complex Systems.*, (1997)
25. Sáez Y., Sanjuán O., Segovia J., Isasi P., "Genetic Algorithms for the Generation of Models with Micropopulations," *Proc. of the EUROGP'03, Univ. of Essex, UK.* Apr. (2003).
26. Kern S., Muller S.D., Hansen N., Buche D., Ocenasek J., Koumoutsakos P. "Learning Probability Distributions in Continuous Evolutionary Algorithms, A Comparative Review"., *Kluwer Academic Publishers*, (2004).
27. Nishio K., Murakami M., Mizutani E., Honda N. "Efficient fuzzy fitness assignment strategies in an interactive genetic algorithm for cartoon face search"., *In Proc. Sixth International Fuzzy Systems Association World Congress (IFSA'95), pp. 173–176*, (2005).
28. Y. Saez and P. Isasi and J. Segovia, "Interactive Evolutionary Computation algorithms applied to solve Rastrigin test functions," *In Proc. of Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST 05), Springer-Verlag, pp. 682–691* (2005).

Creating Chance by New Interactive Evolutionary Computation: Bipartite Graph Based Interactive Genetic Algorithm

Chao-Fu Hong[1], Hsiao-Fang Yang[2], Leuo-hong Wang[1], Mu-Hua Lin[1], Po-Wen Yang[1], and Geng-Sian Lin[1]

[1] Department of Information Management, Aletheia University, NO. 32 Chen-Li St., Tamsui, Taipei County, 25103, Taiwan (R.O.C.)
{cfhong, wanglh, fa925710, aa912588, aa912219}@email.au.edu.tw

[2] Department of Management Information Systems, National Chengchi University, NO. 64, Sec. 2, ZhiNan Rd., Wenshan District, Taipei City 11605, Taiwan (R.O.C)
jimmy52@ms35.hinet.net

Abstract. In this paper, our model supplies designing environment that used the component network to identify the high score components and weak components which decrease the number of components to build a meaningful and easily analysis simple graph. Secondary analysis is the bipartite network as the method for formatting the structure or the structure knowledge. In this step the different clusters' components could link each other, but the linkage could not connect the components on same cluster. Furthermore, some weak ties' components or weak links are emerged by Bipartite Graph based Interactive Genetic Algorithm (BiGIGA) to assemble the creative products for customers. Finally, we investigated two significantly different cases. Case one, the customer did not change his preference, and the Wilcoxon test was used to evaluate the difference between IGA and BiGIGA. The results indicated that our model could correctly and directly capture the customer wanted. Case two, after the Wilcoxon test, it evidenced the lateral transmitting using triad closure extent the conceptual network, which could increase the weight of weak relation and retrieved a good product for the customer. The lateral transmitting did not present its convergent power on evolutionary design, but the lateral transmitting has illustrated that it could quickly discover the customer's favorite value and recombined the creative product.

Keywords: Bipartite, chance discovery, BiGIGA, IEC.

1 Introduction

Kotler and Trias De Bes (2003) at the Lateral Marketing said, "The creativity which in the customer's designing process is a kind of lateral transmitting, and only the customer really knows what he wants." The evolutionary computation must have a subjective (fitness function) for evaluation; thus, it does not suit personal design. The first application of IEC was that it had helped the witness to recognize the criminal face (Caldwell and Johnston, 1991), and it broken the limitation of EC (evolutionary computation) which needed the fitness function to implement the evolutionary computation. Therefore, the interactive genetic algorithm (IGA) (Nishino, Takagi, Cho and Utsumiya, 2001, Ohsaki and Ingu, 1998) has become a useful method. In the EC, the better chromosome has best chance to propagate the genes to the offspring; in contrast with the IEC, the interactive process not only supplies the stimulating information for a designer to drive him discovering what he wants, but also supplies the choosing and recombining power for a designer to make a creative product. Consequently the creative designing problem has become how to help the designer emerging a new creativity or recombining the creative concept into product. In addition, the fatigue issue is the problem of IEC (Nishino, Takagi, Cho and Utsumiya, 2001; Ohsaki and Ingu, 1998; Takagi, 2001), for example, the chromosome contains i genes and each gene has j levels, then there will be j ^ i assembling patterns and its presenting population will be about 6-12. This means that the probability for a designer to find the best pattern is population / (j*i). However, the interactive process has become a heavy load for a designer to find a good solution in the limited timeline. In this paper, we introduce creating chance system which using the text mining technology to discover the weak ties on bipartite network (components network and product linking network) to reduce the customer's fatigue problem.

2 Related Works

2.1 Fitness Function Predicting

One of reducing fatigue problem is the fitness function predicting method. Lee and Chao (1999) used the sparse fitness evaluation which included the clustering technology and fitness allocation method to reduce customer burden. But, in the experiment they used DeJong's five functions as the fitness function; they did not really include the human being into genetic algorithm. Nonaka and Takeuchi (1995) proposed the knowledge growing theory. From the theory we know that on the designing process the customer want change component within chromosome which what he wanted. However, depending on the fitness could not resolve the designing problem. Hayashida and Takagi (2000) proposed the visualized IEC that combined the different capabilities of EC and human beings searching for a global optimum. However, the EC searched in n-D space and the human being globally searched on 2-D map. This means that using SOM distorted n-D to 2-D; then, human being according to the 2-D information decided his favorite position on the map. The EC relied on new fitness function to search the favorite product. In order to assemble the creative product we must understand the whole subjective space (2-D) and n-D context space. However, it seems not easy for a customer to use it. From these papers we know how to recognize the customer's mind is a very important issue, because it not only understands the fitness function but also needs to know what the favorite cluster of component is customer wants. From the past experiment data we discover that the components' association rules can help us understand the customer's favorite cluster or component network. In the next section we want to discuss how to understand what is customer wants and how to create a chance from history data's structural information for him.

2.2 How to Create the Chance

Ohsawa and McBurney (2003) analyzed the contexts to define the high frequency term and mimed the relationship between the terms and then to build the associating graph. Finally, according to the important terms and associational terms they discovered the important weak key. Thus, these terms were as the chance. Watts (2003) proposed bipartite network which was built by multi-identity networks, their relationship were very complex. For example, Watts described the relationship between the actors and movies as shown in Fig.1. After collecting the data, according to the data, Watts built the component affiliation network (as the terms and connection by Ohsawa and McBurney defined) and let assembling like the cluster group network. The network could emerge new chance or creative chance by change the original structure of cluster group network. It means that bipartite network could depend on the overlap cluster network to expand the component network. It means had more flexibility to expand the network for creating the chance too.

The previous discussion indicates that a customer had his favorite components' cluster; hence the system collected the interactive data, and then passed through the associative analyzing we can sketch out the favorite component cluster and associating pattern. After the analysis the triad closure method is used to expand the linking structure and assemble the new product. In this step, we expect that the component network is as the affiliation network; therefore, the product is assembled by some components which structure is as the bipartite network. Finally, according to the component affiliation network and bipartite network build the overlap network. For this reason, we have two kind triad closure processes at least: one is expanding on component network and the other is expanding on overlap network.

From the previous discussion, we think that these methods depended on the network's structural information to discover the chance. However, we use the graph theory to recognize the interactive data in designing process, and analyze the structural information from the characteristic network, which may result in discovering the creative chance.

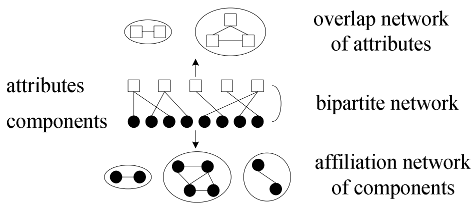

overlap network of attributes
attributes
components
bipartite network
affiliation network of components

Fig. 1. Affiliation network

3 BiGIGA Model - Chance Creating

In this paper our application was what kind of cell phone is customer wanted. Therefore, we used VFT (value-focused thinking) (Keeney, 1992) and brainstorm to define the objective, meaningful and context space. We implemented three brainstorm meetings and every brainstorm meetings includes twenty people attended. The result of brainstorm by VFT is shown on the Table 1 (cell phone's chromosome). The total designing space of cellular phone was 16384 ($2^6 \times 2^2 \times 2^2 \times 2^2 \times 2^2$).

Table 1. The cellular phone's chromosome design

Faceplate	Headset	screen	function-key	number-key
1..64	1..4	1..4	1..4	1..4

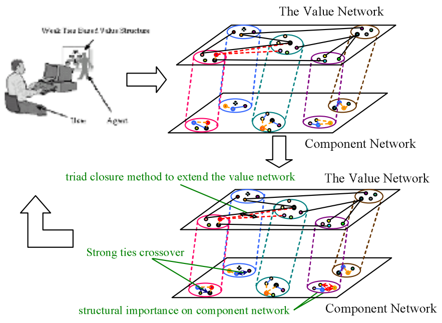

Weak Ties Based Value Structure
User
Agent
The Value Network
Component Network
triad closure method to extend the value network
The Value Network
Strong ties crossover
structural importance on component network
Component Network

Fig. 2. System architecture

Our system as shown in Fig.2, the agent collected the interactive data and then to analyze what components and links can be accepted by customer; then according to the accepted data to discover the objective network and to construct customer's value network. Therefore, the strong ties components is used to building the component network and the strong ties product is used to building the bipartite network. Finally, the triad closure method is used to extend the component network and overlap network for discovering weak ties components (assemble the creative product).

Weak Ties Mining Architecture

As shown in Fig.3 the system generated six products requested the customer to evaluate what product is wanted. After evaluation the system would collect interactive data as presented on table2.

3.1 High Score Key

We defined the columns and rows on the table2 as the sentence, the high score component that was higher than the threshold value would be selected, relatively, the low score component that was lower than the threshold value would be removed. The high score component was defined by Eq.1

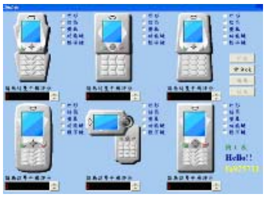


Fig. 3. The interface of BiGIGA

Table 2. Experimental results

	Face.	hand.	screen	fun.	num.	score
1	6	6	6	6	6	9
2	2	2	2	2	2	7
…						
5	7	7	7	7	7	8
6	5	5	5	5	5	3

$$\textbf{If } highscore\_term: a_c > threshold\_value \textbf{ then } a_c = \frac{\sum_{i=1}^{p} C_c}{T_c} \qquad (1)$$

C_c was the score given by the customer for *c* component, T_c was the appearing time of *c* component, and *p* was the population size.

Here, according to the set of high score key *a* (strong ties) to execute the crossover process as shown in Fig.4.

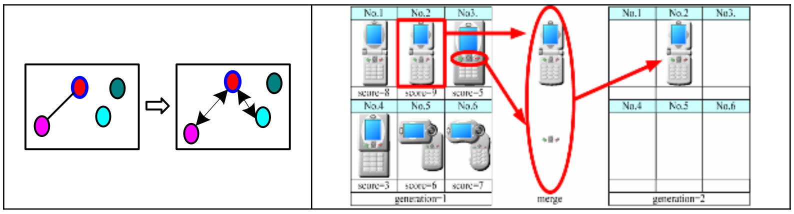


Fig. 4 a. The set of high score key and change each other, b. Mapping to the product

3.2 Discover the Weak Ties as the Chance on Component's Network

According to the interactive data, we can know the relationship between components and the important key *key (a)* (by Eq.2). Here we added an environmental constraint: only to count that in the same cluster's link. It using Eq.1 and Eq.2 to sketch out the KeyGraph (the strong component not only is the important keys, but also is the high score key) (as shown in Fig.5). It uses Eq.2, the components which are the weak components (not that high score components, but that the important components). The solid line represents the strong link (same cluster), and the dotted lines are the weak link (different cluster) (as defined by Eq.3). These weak ties components and weak ties links' components are as the bridge (chance) which can guide the old network to the new creative network.

$$
\begin{array}{ll}
Assoc(a_i, a_j) = \min\left(|a_i|_s, |a_j|_s\right) & key(a) = 1 - \prod_{g \subset G}\left[1 - \frac{base(a,g)}{neighbors(g)}\right] \\
base(a,g) = |a|_s |g-c|_s & neighbors = \sum_{a \in s} |a|_s |g-a|_s \\
if\ a \in g,\ \ |g-a|_s = |g-a|_s & if\ a \notin g,\ \ |g-a|_s = |g|_s
\end{array}
\tag{2}
$$

The *s* is the sentence, the *g* is the cluster, *D* is the sentence, also is the article, the *G* is the all clusters.

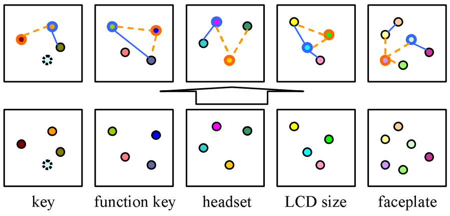


Fig. 5. Component's value network

After the previous procedures, using triad closure method to extend the component network (from dotted to solid), and the operation is shown in Fig.6.

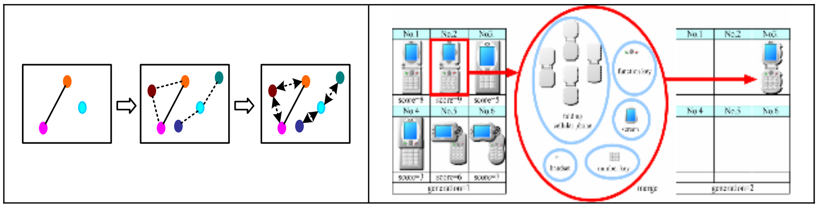


Fig. 6 a. The triad closure method, b. Mapping to the product

3.3 Discover the Weak Ties as the Chance on Product Network

After analyzing the important and weak components, in this section we want to understand how to assemble each other.

$$
\begin{array}{ll}
Assoc(a_i, a_j) = \min\left(|a_i|_{pis}, |a_j|_{pjs}\right) & for \quad pi \neq pj \\
key(a) = \sum_{j=1}^{n} Assoc(a_i, a_j) & for \quad i \neq j
\end{array}
\tag{3}
$$

The a is the set of components, the pi and the pj are the part i and part j, and n is the number of selected components. It depended on the distance (link's weight) to separate it to the strong link or weak link. Of course, the link crossover was that we want changed to assemble the potential products (as shown in Fig.7).

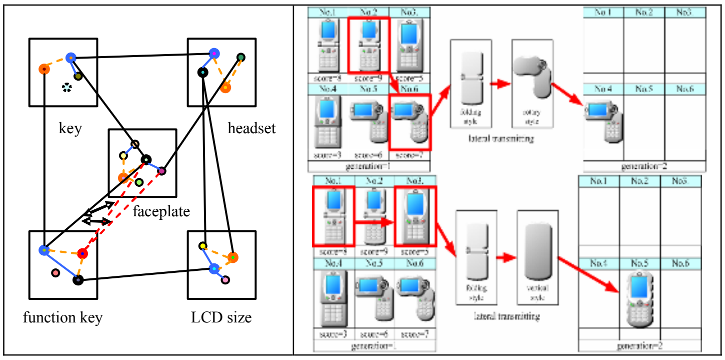


Fig. 7 a. Triad closure method to extend the product network, b. Mapping to the product

3.4 The Crossover Processing - Chance Creating

The *a* is the weak ties components and weak ties links' components, the *b* is all the high score component set and *b'* is the partial of *b*.

$$Crossover = a_t + b'_t \tag{4}$$

However, the partial of *b* is replaced by *a* to make a creative product *a*+*b'*, that is as the chance creating operation.

In this operation, it includes the strong ties crossover that strong ties components or strong links replaced each other and the weak ties crossover that weak ties components replace the strong ties components or the weak links replaced the strong links. These mechanisms are not like the IGA (depends on the strong ties crossover) have a little chance for mutation (the strong components replaced by weak ties components) and less triad closure expanding method creates the chance.

3.5 Mutation

In VFT process we could easily get the customers' fundamental objectives, meaningful and context space. Then, according to different requirements we design the value network and lock some favorite components and links' structure what customer wants. Beside this, the mutation is very important process that it could generate the diversity of components or links in the evolutionary process. It will be a chance for searching his favorite component network and value network (as shown in Fig. 8).

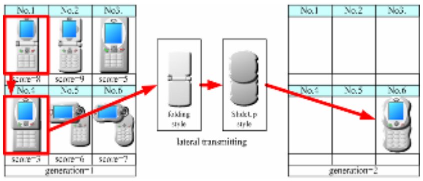


Fig. 8. Mutating operation

4 Experimental Design

The purpose of this experiment was that recombined the weak ties and bipartite network with IEC to guide the customer design his favorite product. First, we had surveyed the cellular-phone's market and implemented three brainstorm meetings to design the customer's value network and the context of the cellular-phone. The cellular-phone was assembled by five parts as the faceplate, handset, screen, function-key and number-key. In our experiment, the faceplate had four types and each type had sixteen levels, the handset had four levels, the screen had four levels, the function-key had four levels and the number-key had four levels. The design of parameters of IGA and BiGIGA were shown on Table 3.

Table 3. Parameter's setting

	IGA	BiGIGA
coding	binary	binary
selection method	elitism	elitism
population size	6	6
crossover method	one-point	Lateral transmitting (near)
crossover rate	0.8	×
mutation method	one-point	lateral transmitting (far)
mutation rate	0.01	×

Twenty-two samples were drawn from the professors and colleges of university in north Taiwan. After the experiment, we had celebrated some tests to evidence the BiGIGA had the creative ability better than the IGA.

5 Case Study

5.1 The Variation of Prefer Components

In order to compared the variation of prefer components between the IGA and the BiGIGA, we defined the Eq.5 for calculating the entropy, to investigate the variation of prefer components.

$$entropy = \frac{appeared\_component}{total\_component} \tag{5}$$

The appeared_compponent is customer prefer components.

The results of analysis data are shown in Fig. 9, it has two significant cases, 4 (case 1) and 16 (case 2).

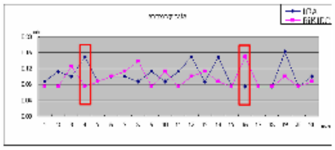


Fig. 9. Entropy rate analysis

5.2 Case 1 –Customer Who Had No Lateral Transformation

In the case 1, the analysis result is shown in Fig. 10. At g=1 all products were generated by random. At g=3 for IGA, the strong ties components dominated the evolutionary and as a result of the customer can quickly found his prefer product, but for BiGIGA its natural mechanism must mix the strong ties and weak ties on searching processes to help customer design, therefore, after the Wilcoxon matched-pairs signed-ranks test (the IGA was 0.05 and the BiGIGA was 0.10), the result indicated that if the customer really knew what he wants BiGIGA model would work as same as the IGA or little worse than IGA.

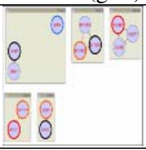
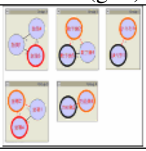
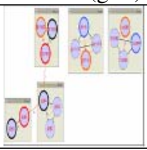
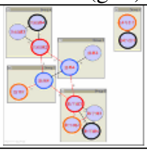
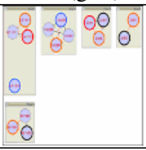
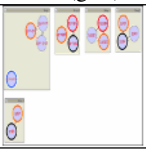
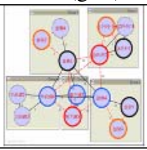
BiGIGA(g=1)	BiGIGA(g=2)	BiGIGA(g=3)	BiGIGA(g=4)
			
IGA(g=1)	IGA(g=2)	IGA(g=3)	
			

Fig. 10. Case 1

5.3 Case 2 –Consumer Who Had Lateral Transformation

At g=1, BiGIGA lacked enough data to build the product network. At g=2, BiGIGA found some good products and obviously built the product network. After g=2, the BiGIGA brought the lateral transmission and influenced the customer's value structure. The valid data quickly decreased it brought the broken product network; the graph was as the Small World on critical state. The weak ties components as the short-cut on Small World, at g=4, the high density product network was built. After the Wilcoxon matched-pairs signed-ranks test (IGA is 0.0017, BiGIGA is 0.09), which means that the BiGIGA's components' variation was higher than the IGA. From Fig.11 BiGIGA, we clearly observed that the customer was influenced by weak components and shifted his favorite to other cluster for creating innovation. And then, the new components' cluster would supply for customer to continue his design, enlarging the entropy. When the valid products were enough to build the product network, the strong link connected with strong components for assembling the strong products. The IGA depend on the crossover and mutation to assemble the favorite product (time consuming as a result of lack chance to change), relatively, the BiGIGA has three kinds weak ties mechanism that owns big chance to change. It evidenced our bipartite graph could response the lateral transmission and according to the short-cut quickly discovered the favorite product.

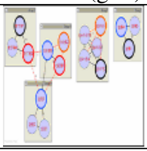
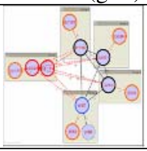
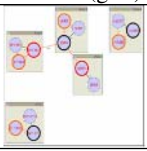
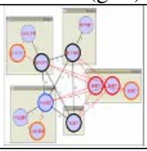
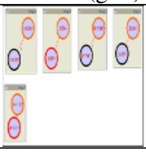
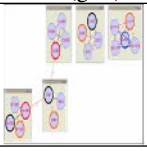
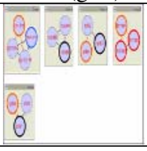
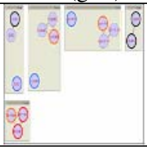
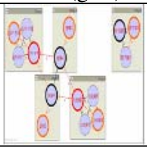
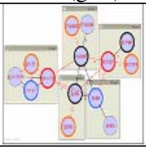
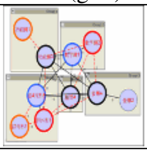
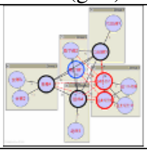
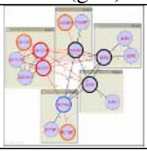
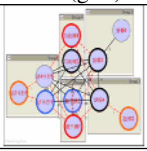
BiGIGA(g=1)	BiGIGA(g=2)	BiGIGA(g=3)	BiGIGA(g=4)	BiGIGA(g=5)
				
IGA(g=1)	IGA(g=2)	IGA(g=3)	IGA(g=4)	IGA(g=5)
				
IGA(g=6)	IGA(g=7)	IGA(g=8)	IGA(g=9)	
				

Fig. 11. Case 2

6 Conclusions

In this study, we developed a creative designing system and it supplied three kinds of weak ties to extend the complexity of network for emerging the chance and helped the customer to design his favorite product. In addition, the BiGIGA is not as same as the IGA, which not only depends on the nature selection (strong ties), but also depends on the weak ties that can keep the diverse evolution. Such the mechanism may significantly decrease the customer's fatigue problem. The experimental results also indicate that the BiGIGA not only to calculate the component's weight, but also relied on the bipartite network to expand the weak ties for increasing the diversity. Such the mechanism can quickly expand the useful network and discover the favorite product for the customer (average times: IGA is 10, BiGIGA is 6.31). Beside this we investigated the graph, when the graph became more complex and the diameter of cluster quickly decreased and become a Small World. In the Small World environment the customer could easily design diverse products and discover the product that he wanted. This is a very interesting weak ties phenomenon.

References

Caldwell, C., and Johnston, V. S., (1991), "Tracking a criminal suspect through 'face-space' with a genetic algorithm," Proceedings of the fourth international conference on genetic algorithms, San Francisco, CA: Morgan Kauffman Publishers Inc, pp. 416-421.

Hayashida, N., and Takagi, H., (2000), "Visualized IEC: Interactive Evolutionary Computation with Multidimensional Data Visualization," IEEE International Conference on Industrial Electronics, Control and Instrumentation (IECON2000), Nagoya, Japan, pp.2738-2743.

Keeney, R. L., (1992), "Value focused thinking: A path to creative decision making," Cambridge, MA: Harvard University Press.

Kotler, P., and Trias De Bes, F., (2003), "Lateral marketing: New techniques for finding breakthrough ideas," John Wiley & Sons Inc.

Lee, J. Y., and Cho, S. B., (1999), "Sparse fitness evaluation for reducing user burden in interactive genetic algorithm," IEEE International Fuzzy Systems Conference Proceedings, pp. 998-1003.

Nishino, H., Takagi, H., Cho, S., and Utsumiya, K., (2001), "A 3d modeling system for creative design," The 15th international conference on information networking, Beppu, Japan, IEEE Press, pp. 479-486.

Nonaka, I. and Takeuchi H., (1995), "The knowledge-Creating Company." Oxford Press.

Ohsaki, M., Takagi, H., and Ingu, T., (1998), "Methods to reduce the human burden of interactive evolutionary computation," Asia fuzzy system symposium, Masan, Korea, IEEE Press, pp. 495-500.

Ohsawa, Y., and McBurney, P., (2003), "Chance Discovery," New York: Springer Verlag.

Takagi, H., (2001), "Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation," Proceeding of the IEEE, IEEE Press, pp. 1275-1296.

Watts, D. J., (2003), "Six Degrees: The Science of a Connected Age," Norton, New York.

Interactive Evolutionary Computation Framework and the On-Chance Operator for Product Design

Leuo-hong Wang[1], Meng-yuan Sung[2], and Chao-fu Hong[1]

[1] Evolutionary Computation Laboratory, Department of Information Management, Aletheia University, Taiwan
{wanglh, cfhong}@email.au.edu.tw
[2] Graduate School of Management Sciences, Aletheia University, Taiwan
aa902728@email.au.edu.tw

Abstract. Traditionally, product design problem is usually solved by means of the conjoint analysis methods. However, the conjoint analysis methods suffer from evaluation fatigue. An interactive evolutionary computation (IEC) framework for product design has been thus proposed in this paper. The prediction module taking care of evaluation fatigue is the main part of this framework. In addition, since the evaluation function of product design is an additive utility function, designing operators which heavily utilizes the prediction results becomes possible. The on-chance operator is thus defined in this paper as well. The experimental results indicated the on-chance operator can speed up IEC and improve the quality of solution at the same time.

1 Introduction

Product design is one of the most important tasks of product development process [1]. Traditionally, in product design phase, a product is represented by a vector of *attributes* referring to customer needs and product specifications (for example, appearance, price, functionality and so forth). Moreover, each attribute includes several different alternatives named as *attribute levels*. Given a vector of attributes, conjoint analysis [2][3], which is an interactive and structural technique, is the most common used approach to determine the optimal product. However, as proven in [4], product design is a combinatorial optimization problem and hence NP-hard. Once the number of attributes or levels is very large, the interactive process of conjoint analysis becomes infeasible because of evaluation fatigue. Various approaches have therefore been proposed for solving product design problems with large amounts of attributes or levels, including adaptive conjoint analysis [5] and polyhedral adaptive conjoint [6]. In this paper, we propose an interaction evolutionary computation (IEC) framework to address the product design problem since IEC is a powerful tool for identifying the user preference [7].

Although being capable of identifying the user preference, IEC suffers from evaluation/ human fatigue as well. Therefore, the canonical IEC is intuitively insufficient for addressing the product design problem. In other words, the IEC approach for product design will work out only if the human fatigue problem of IEC is appropriately addressed. A prediction module, which learns user evaluation patterns for predicting fitness values of succeeding individuals, is thus incorporated into our IEC framework for product design. Moreover, due to the evaluation criterion for product design has an additive functional structure, which is actually a *utility function* found in utility theory [8] , the learning results can be further used to design genetic operators for accelerating the searching process. As a result, we propose an IEC with prediction module framework and a specific genetic operator named as *on-chance* operator, which is different from the traditional *by-chance* genetic operators, for product design in this paper.

The rest of the paper is organized as follows. Section 2 begins with giving the formal definition of the product design problem. IEC framework for product design, which incorporates with a prediction module, is then presented. The idea of on-chance operator is also introduced in this section. Section 3 illustrates our experimental design first. Actually, we implemented an IEC without prediction for comparison. Then experimental results are shown in this section and followed by a discussion on results. The final section presents the conclusion and outlines our future work.

2 The Problem and the IEC Framework

2.1 The Product Design Problem

As mentioned earlier, the conjoint analysis model is the most common approach to address the product design problem. The problem will be formulated as a combinatorial optimization problem under conjoint analysis model. In other words, a product is considered to be a vector of k relevant attributes. And further, each attribute $a_i (i = 1, 2, \cdots, k)$ has s_i different attribute levels, $l_1^i, \cdots, l_{s_i}^i$, for instance. The product design problem therefore becomes a problem of searching the optimal combination of attribute levels to satisfy the user preference. Assume P is the set of all possible *product profiles* (a *product profile* is an arbitrary combination of attribute levels):

$$P = \{a_1 a_2 \cdots a_k | a_i \in \{l_1^i, l_2^i, \cdots, l_{s_i}^i\}\} \tag{1}$$

and $F(p)$ is the user preference or evaluation function:

$$F(p) = \texttt{the score that the user gave to } p, \forall p \in P, \tag{2}$$

then searching the optimal product profiles will be equivalent to determine $\exists p^* \in P$ such that $F(p^*) \geq F(p), \forall p \in P$ and $p \neq p^*$.

Based on equations 1 and 2, product design is extremely easy if the number of possible profiles is small enough. However, the real world applications usually have so many attributes or levels that evaluating all possible profiles is infeasible. Conjoint analysis model therefore assumes that the evaluation function is a utility function, which is a linear combination of part-worth utility value of each attribute level:

$$F(p) = \sum_{i=1}^{k} w_i \cdot U(a_i) \tag{3}$$

where p is a specific profile whose ingredients are $a_1, a_2, \cdots a_k$, and a_i equals to one of its attribute levels $l_1^i, \cdots, l_{s_i}^i$; $U(a_i)$ is the utility function of a_i. Given l_j^i, which is one of the attribute levels of a_i, $U(a_i = l_j^i)$ will return the part-worth utility of l_j^i. Eventually, w_i is the weighted coefficient of $U(a_i)$.

Under the assumption mentioned above, multivariate techniques can be applied to figure out the optimal product if the user has evaluated sufficient amounts of product profiles generated by some experimental design techniques, such as orthogonal arrays [9]. Unfortunately, if the minimal number of product profiles required to be evaluated for identification of optimal solution is very large, the evaluation fatigue will be arisen and consequently bias the analytic results. Other researches mentioned in Section 1, such as adaptive conjoint and polyhedral adaptive conjoint, had tried to decrease the number of evaluation needed for getting the optimal solution. But as we know, none of researches addressed this problem by using IEC approach. That motivated us to propose an IEC framework for product design in this paper. However, dealing with human fatigue is the most important issue for this framework.

2.2 The IEC Framework

Incorporating strategies to deal with human fatigue is inevitable to our IEC framework for product design. Takagi, in his IEC survey [7], listed several strategies for easing user burden to conquer the human fatigue problem of IEC. Recently, Saez et. al, proposed a class of IEC framework, which named as Fitness Predictive Genetic Algorithm (FPGA), to address human fatigue problem [10][11]. Llorà et. al., proposed a multi-objective approach using support vector machine (SVM) as the learning method to conquer user fatigue [12]. Both of them claimed fewer generations are needed than the other methods. However, if we follow the conjoint analysis model to formulate the product design problem in our IEC framework, then the strategy of predicting fitness values via learning user evaluation patterns will be more suitable than the other strategies since equation 3 precisely defines the target function for learning. Therefore, a prediction module, which learns user evaluation patterns, is incorporated into our IEC framework in order to reduce user burden.

The learning methods that have been proposed for various IEC applications were mainly the neural networks (NN) [13][14][15][16]. Nevertheless, no matter what methods are applied, the user burden will be reduced more or less because fewer generations will be needed for finding out *satisficing solutions* [17], which are good enough solutions for users. In order to learn the utility function defined by equation 3 from IEC evaluation data, a genetic algorithm (GA) instead of the commonly used NN was chosen to serve as the core of the prediction module.

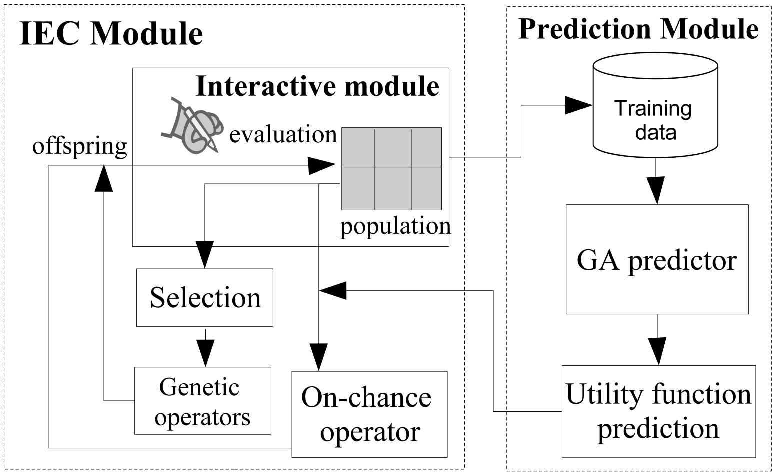


Fig. 1. The system architecture of our IEC framework

Easier implementation is the main reason for choosing GA. Fig. 1 illustrates the system architecture of our IEC framework for product design.

Two primary modules, including an IEC module and a prediction module, are sketched in Fig. 1. The IEC module is responsible for interacting with the user generation by generation and eventually evolving satisficing solutions for the user. However, evaluation data will be fitted into the prediction module for learning user evaluation patterns. More importantly, an *on-chance operator* is defined according to the feedback from the prediction module. The on-chance operator is a deterministic operator, which replaces bad genes, or bad attribute levels equivalently, with better ones. The offsprings generated by the on-chance operator will be combined with other offsprings produced by canonical genetic operators to form the next generation. The details concerning the on-chance operator will be explained in the Section 2.3.

The prediction module is actually a GA predictor. The evaluation data collected by the IEC module will serve as the training data for the GA predictor. The chromosome of the GA predictor is an array of the prediction values of all attribute levels. For example, if a product consists of k relevant attributes and each attribute a_i has s_i attribute levels, then the chromosome will be an array shown in Fig. 2.

In Fig. 2, all k attributes from a_1 to a_k are included in this array. Moreover, each element of the array represents the prediction utility value of the corre-

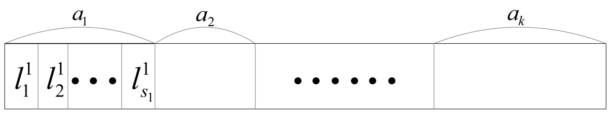


Fig. 2. The chromosome of the GA predictor

sponding attribute level. As such, the root mean square (RMS) error function is an intuitive way to compute the fitness value of specific chromosome. The fitness function of the GA predictor is thus defined as follows:

$$F(c) = \sqrt{\frac{\sum_{i=1}^{n}(\hat{u}_i - u_i)^2}{n}} \tag{4}$$

where c is a chromosome, n is the number of training data, u_i is the utility value of i^{th} training example, which is actually the score given by user and $\hat{u}_i$ is the utility value of i^{th} training example predicting by the chromosome c.

The GA predictor will minimize the RMS error as much as possible and generate a utility prediction function for feeding back to IEC module. As a result, this function can be utilized to evaluate the succeeding individuals as well as to design the on-chance operator.

2.3 The On-Chance Operator

The genetic algorithm uses random choice as a tool to guide the searching process [18, page 5]. Such a guided random search works fine if execution time or evolving generations is unlimited. However, random choice sometimes is unsuitable for IEC since IEC is strongly constrained by execution time. Therefore, if a searching direction is determined by the prediction function, evolving individuals along with this direction is an approach worthy to be considered.

The prediction module mentioned in Section 2.2 will identify a utility prediction function after each generation of IEC. Using this prediction function, the optimal product profile is easy to find because the additive structure of utility function. In other words, the product profile consisting of attribute levels with highest prediction values will be the optimal solution if the prediction function is correct. Therefore, we define an *on-chance* operator which heavily utilizes the prediction results for evolution of the IEC population.

The on-chance operator manipulates chromosomes by the following ways:

1. **Gradient ascent.** Replacing the bad genes, which are attribute levels with lowest prediction utility values, by the best ones.
2. **Emergence.** Replacing the bad genes by the attribute levels which never appear before or appear fewer times than the others. The prediction values of attribute levels that never appear before are obviously error-prone. Moreover, experimental design techniques, such as orthogonal arrays, usually suggest that each attribute level should appear equally for estimating utility values correctly. The reason to emerge such attribute levels is therefore all concerning the prediction correctness.
3. **Elitism.** The optimal product profile predicted by the prediction function will be chosen directly into the next generation.

As a result, both the on-chance operator and the canonical genetic operators, which are typically by chance, are used to evolve the population of IEC.

3 Experiments and Discussion

3.1 Experimental Design

Two systems, including a canonical IEC ($cIEC$) and the IEC framework (IEC_{oc}, oc stands for on-chance) presented in Section 2.2, have been implemented for comparison. The main issues are the efficiency and the quality of solution. Moreover, we have tried various combinations of the on-chance operator and canonical genetic operators to investigate how the on-chance operator impacts on the evolution of IEC.

A cellular phone design application is selected for investigation. More precisely, a cellular phone consists of five different attributes in this research. The attributes includes *appearance, display, numerical keypad, receiver* and *control button*. The appearance attribute has 19 different attribute levels and the others have 4 attribute levels respectively. Fig. 3 illustrates all these 35 attribute levels.

Some important attributes of a real world product design application (such as price, for instance) are missing in the specification listed above. However, the combinatorial nature is preserved in this test application.

The population size of both $cIEC$ and IEC_{oc} were 9. Two-way deterministic tournament selection was used for both IEC systems. $cIEC$ adopted single point crossover and mutation with probabilities 80% and 0.6% respectively. Ten-point scale was used for users to evaluate each candidate cellular phone. If the user gave any one cellular phone 10 points, $cIEC$ or IEC_{oc} would terminate. Then the cellular phone getting 10 points would be the final solution. Fig. 4 shows the interface of the IEC_{oc}.

A series of Monte Carlo simulations had been running to determine the parameter settings of the GA predictor mentioned earlier. The parameter settings are listed in Table 1.

Eventually, the laboratory experiment was used to evaluate the performance. Dozens of subjects were invited to manipulate IEC systems to identify their most

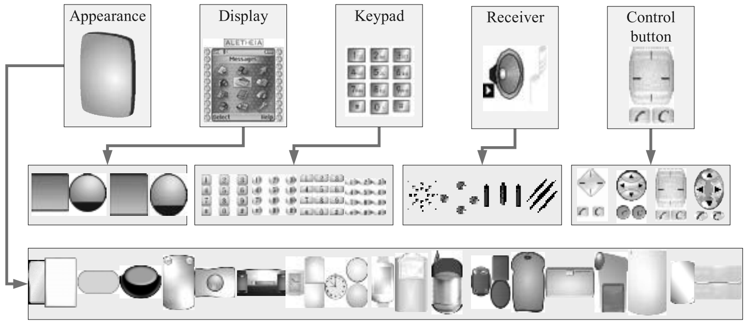


Fig. 3. The cellular phone design application. Five attributes, including appearance, display, numerical keypad, receiver and control button are used in this application. The appearance has 19 attribute levels and the others have 4 attribute levels respectively.

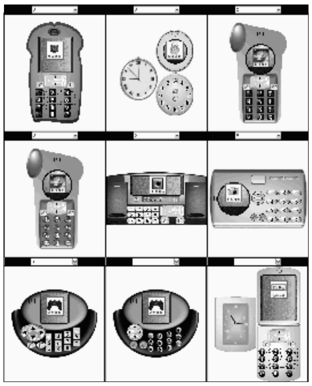


Fig. 4. The user interface of IEC_{oc}

Table 1. The parameter settings of the GA predictor

population size	= 20
crossover rate	= 0.95
mutation rate	= 0.14
number of generations	= 500 or 80% of individuals dominated by elitist

favorite cell phones. The subjects were randomly divided into several groups and were requested to operate one of the systems.

3.2 Experimental Results and Discussion

To understand the impact of the on-chance operator, several individuals of current generation, from 1 to 4, were selected and manipulated by the on-chance operator. The remaining individuals were manipulated as usual. We called these 4 different combinations of the on-chance operator with canonical genetic operators as IEC_{oc-1}, IEC_{oc-2}, IEC_{oc-3}, and IEC_{oc-4} respectively. IEC_{oc-1} randomly selected one individual from current population and replaced it with the optimal product profile predicted by the GA predictor. Except the elitism, IEC_{oc-2}, IEC_{oc-3} and IEC_{oc-4} further replaced individuals based on the other rules: the gradient ascent and emergence mentioned in Section 2.3. However, the emergence rule had higher priority.

The number of generation and the execution time. The average numbers of generation needed by five different IEC systems are listed in Table 2. However, because each system was only manipulated by 5 subjects, Mann-Whitney test, which is a non-parametric test procedure was used to investigate the significance of experimental results. As a result, the data listed in Table 3 are yielded by the test procedure.

In Table 3, the average numbers of generation of IEC_{oc-1} and IEC_{oc-4} are both significant fewer than *cIEC*. The other two versions of IEC_{oc} need less

Table 2. The number of generation needed by IEC systems

	$cIEC$	IEC_{oc-1}	IEC_{oc-2}	IEC_{oc-3}	IEC_{oc-4}
average number of generation	11.4	6.2	8.2	8.2	6.8

Table 3. Mann-Whitney test results for number of generations executed by several IEC systems. z is the observed z value of the standard normal distribution. Then p is the proportion of the normal distribution falling to the right of the z value. Therefore, if p is less than 0.05 or even 0.01,then the result is significant.

	IEC_{oc-1}		IEC_{oc-2}		IEC_{oc-3}		IEC_{oc-4}	
	z	p	z	p	z	p	z	p
$cIEC$	2.09	0.0183	1.46	0.0721	1.46	0.0721	2.09	0.0183
IEC_{oc-1}	–		-1.15	0.1251	-0.94	0.1736	-0.52	0.3015
IEC_{oc-2}	–		–		0.1	0.4602	0.94	0.1736
IEC_{oc-3}	–		–		–		0.52	0.3015

Table 4. Mann-Whitney test results of satisfaction investigation for *cIEC* and various IEC_{oc}

	IEC_{oc-1}		IEC_{oc-2}		IEC_{oc-3}		IEC_{oc-4}	
	z	p	z	p	z	p	z	p
$cIEC$	-4.71	<.0001	-4.79	<.0001	-4.95	<.0001	-4.39	<.0001
IEC_{oc-1}	–		-1.37	0.0853	-1.4	0.0808	1.12	0.1314
IEC_{oc-2}	–		–		0.0	0.5	2.74	0.0031
IEC_{oc-3}	–		–		–		2.47	0.0068

generations than *cIEC* too, though the results are not significant. Therefore, IEC_{oc} is no doubt faster than *cIEC* for the most cases. That implies the on-chance operator speeds up the convergence of IEC.

Satisfaction investigation. After manipulating the IEC systems, each subject was further asked to fill in a questionnaire which included topics related to the appearance, functionalities and desire to purchase of the derived cell phones. The other questionnaire containing questions on innovation and correctness was also necessary to fill. Five-point scale, with *1="very good"* and *5="very bad"* was used for both questionnaires. Table 4 lists the statistical test results.

All four versions of IEC_{oc} significantly outperform *cIEC* in the satisfaction investigation. Recall from Table 3, IEC_{oc} needs fewer generations than *cIEC* for finding out solution, we can claim the on-chance operator can accelerate the convergence of IEC and improve the quality of solution as well. However, it is worth to note that IEC_{oc-2} or IEC_{oc-3} seems outperforming the other two versions.

Comparison of IEC_{oc} with other algorithms for product design. The orthogonal array technique is one of the most important methods to design product profiles for evaluation. However, a minimal design is very difficult to find for a complicated product whose attributes or attribute levels are plenty. We can just estimate the number of profiles needed to evaluate for our cellular phone design application from the literature. As a result, no more than 80 product profiles will be needed for our design application, since a minimal design for a problem with 5 attributes and their attribute levels are 20, 8, 8, 8 and 8, respectively, whose problem size is larger than ours, actually need only 80 profiles [9]. In other words, if the population size of IEC is 9, then 9 generations are the upper bound for finding final solutions. Fortunately, IEC_{oc} almost needs 6 to 8 generations to find out solutions according to the experimental results listed in Table 2.

However, new algorithms for product design presented recently, especially the fast polyhedra conjoint, are more effective than the orthogonal array technique. For instance, the fast polyhedra conjoint [6], which is a branch-and-bound technique, can solve our cellular phone design application only by evaluating 35 product profiles (35 is the number of total attribute levels), which is equivalent to 4 generations.

Nevertheless, the IEC approach for product design still has some advantages over algorithms using conjoint model. The learning capability is one of the advantages. Conjoint model assumes the attributes are mutually independent. If attributes are not mutual independent, that is the problem is a non-linear problem, then the profiles needed to evaluate will increase tremendously. However, learning methods such as neural networks and genetic programming can handle the non-linearity easily.

4 Conclusions and Future Work

We have proposed an IEC framework with a GA prediction module for product design. Since using conjoint analysis model to formulate the product design problem, the prediction module can be further utilized. As a result, we have designed an on-chance operator which deterministically manipulates the chromosome according to the prediction results. Experimental results indicated the on-chance operator could speed up the canonical IEC and improve the quality of solution at the same time. This result is promising.

However, as a tool for product design, our IEC framework still needs to improve because the performance is worse than some recently developed algorithms, such as fast polyhedra conjoint. Therefore, how to accelerate IEC further by utilizing the on-chance operator is our short-term goal. Besides, applying other learning methods which can handle the non-linearity among attributes is another research direction.

References

1. Krishnan, V., Ulrich, K.T.: Product development decisions: A review of the literature. Manage. Sci. **47** (2001) 1–21
2. Shocker, A.D., Srinivasan, V.: Multiattribute approaches for product concept evaluation and generation: A critical review. Journal of Marketing Research **16** (1979) 158–180

3. Green, P.E., Srinivasan, V.: Conjoint analysis in marketing: New developments with implications for research and practice. Journal of Marketing **54** (1990) 3–19
4. Kohli, R., Krishnamurti, R.: Optimal product design using conjoint analysis: computational complexity and algorithms. European Journal of Operational Research **40** (1989) 186–195
5. Johnson, R.M.: Adaptive conjoint analysis. In: Sawtooth Software Conference Proceedings. (1987) 253–265
6. Toubia, O., Simester, D., Hauser, J., Daha, E.: Fast polyhedral adaptive conjoint estimation. Marketing Science **22** (2003) 273–303
7. Takagi, H.: Interactive evolutionary computation: Fusion of the capacities of ec optimization and human evaluation. Proceedings of the IEEE **89** (2001) 1275–1296
8. Keeney, R.L., Raifa, H.: Decisions with multiple objectives: preferences and value tradeoffs. John Wiley, New York (1976)
9. Hedayat, A., Sloane, N., Stufken, J.: Orthogonal Arrays: Theory and Application. Springer-Verlag, New York (1999)
10. Saez, Y., Isasi, P., J., S., Hernandez, J.C.: Reference chromosome to overcome user fatigue in iec. New Generation Computing **23** (2005) 129–142
11. Saez, Y., Isasi, P., Segovia, J.: Interactive Evolutionary Computation algorithms applied to solve Rastrigin test functions. In: Soft Computing as Transdisciplinary Science and Technology. Springer-Verlag (2005) 682–691
12. Llorà, X., Sastry, K., Goldberg, D., Gupta, A., Lakshmi, L.: Combating user fatigue in igas: Partial ordering, support vector machines, and synthetic fitness. In: ACM Genetic and Evolutionary Computation Conference (GECCO 2005), ACM press (2005) 1363–1371
13. Biles, J.A., Anderson, P.G., Loggi, L.W.: Neural network fitness functions for a musical iga. In: International ICSC Symposium on Intelligent Industrial Automation and Soft Computing. (1996)
14. Burton, A., Vladimirova, T.: Genetic algorithm utilising neural network fitness evaluation for musical composition. In G.D. Smith, N.S., Albrecht, R., eds.: Proceedings of the 1997 International Conference on Artificial Neural Networks and Genetic Algorithms. (1997) 220–224
15. Johanson, B., Poli, R.: GP-music: An interactive genetic programming system for music generation with automated fitness raters. In: Genetic Programming 1998: Proceedings of the Third Annual Conference, Morgan Kaufmann (1998) 181–186
16. Dozier, G.V.: Evolving robot behavior via interactive evolutionary computation: from real-world to simulation. In: Proceedings of the 2001 ACM Symposium on Applied Computing (SAC). (2001) 340–344
17. Simon, H.A.: The New Science of Management Decision. Prentice Hall PTR, Upper Saddle River, NJ, USA (1977)
18. Goldberg, D.E.: Genetic Algorithms in Search, Optimization&Machine Learning. Addison Wesley (1989)

Practically Applying Interactive Genetic Algorithms to Customers' Designs on a Customizable C2C Framework: Entrusting Select Operations to IGA Users

Fang-Cheng Hsu[1] and Ming-Hsiang Hung[2]

[1] Department of Information Management, Aletheia University
http://fanechi.myweb.hinet.net/
fanechi@email.au.edu.tw
[2] Graduate School of Management Science, Aletheia University
s1788124@email.au.edu.tw

Abstract. We propose a customizable C2C (customer to customer) framework to fully utilize interactive genetic algorithms (IGA) and to discover the potential capabilities of IGAs in customer designs. Traditionally, IGA users assign fitness to each chromosome. No matter the rating or ranking of the assignments, the traditional methods were unnatural, especially when IGAs were applied to customers' designs. In this study, we find that allowing IGA users to directly select chromosomes into the mating pool according to their hidden fitness function(s) is not only a natural way to implement the select operations of IGA, but is also more effective. We call the model where parts of select operations are manipulated by users, the SIGA model. Preventing fatigue, however, is the most important challenge in IGA. The OIGA (Over-sampling IGA) model has been extremely effective at decreasing user fatigue. To verify the performance of the proposed SIGA, we conduct a case study and use the OIGA model as a benchmark. The results of the case study show that the proposed SIGA model is significantly more effective than the IOGA model.

1 Introduction

Fulfilling customers' true needs is essential in marketing. In the current "manufacturers-stores-buyers" commerce framework, buyers must go to stores or shopping malls to choose the products that satisfy their wants. Customers usually spend significant amounts of time looking for their favorite products in the stores, but the items they actually get do not always meet their needs.

In traditional manufacturer-centered designs, market researchers are sent to discover customers' needs. Manufacturers design new products based on researchers' reports. No matter how many "unmet needs" are reported, the manufacturers decide which ideas to develop and assign them to project development teams. Transferring the customers' needs and wants this way might result in biased information and cause the product designers to create unneeded products. Furthermore, this can result in producing low-demand products.

To reduce misunderstandings, manufacturers are investigating the concept of user-centered designs, or customers' designs. Most modern user-centered approaches are where "customers interactively design with manufacturers", and are applicable to B2C (business-to-consumer) and B2B (business-to-business) e-commerce.

The "customers interactively design with manufacturers" strategy is not suitable for personalization, however, and might not be appropriate in a quickly changing environment. A more accurate method of meeting customers' needs is to allow them to design products completely by themselves and make the products accordingly. Executing this complete customers' designs concept on B2C or B2B e-commerce is impractical, except for manufacturers that are well equipped with enormous adaptable and efficient manufacturing processes for various products.

One special feature of C2C (customer to customer) e-commerce is that the buyer's needs are quickly transmitted to an enormous number of individual sellers. This feature makes C2C e-commerce an attractive candidate for developing a framework to accomplish the goal of complete customer design.

In this paper, we propose an Interactive Genetic Algorithms (IGA) based framework to accomplish complete customers' designs on a customizable C2C e-commerce platform. The framework supports customers (buyers) designing products by themselves with IGA-based toolkits equipped by the C2C market provider, i.e. intermediaries of C2C e-commerce, and the products are provided by other customers (sellers) according to their designs. Critical to the framework's success is whether customers accept the IGA-based toolkits. So far, researchers have found evidence supporting the hypothesis that using IGA can create satisfying products within a large alternative space, but the user fatigue problem in IGA still requires a solution [1].

This study not only focuses on providing practical IGA-based toolkits for a customizable C2C framework, but also on assuring that the toolkits is used naturally and reduces fatigue suffered on the customizable C2C framework. We propose a manipulated IGA-based toolkit, and by entrusting select operations of IGA to customers, ensure that the toolkit is practical. To verify the performance of the proposed strategy, we conduct a case study on mineral water bottle design and analyze the differences between the proposed model and a benchmark model.

2 Background

2.1 Manufacturer-Centered Designs and User-Centered Designs

Although Internet stores have provided customers with convenient shopping alternatives and have significantly shortened the shopping cycle, what customers actually get through these stores (shopping malls/Internet stores) does not always meet their needs. An industry's product designers are talented in designing impressive products, but products should not just be impressive. They should meet customers' needs and wants.

To overcome the previously mentioned drawbacks, researchers proposed the concept of user-centered designs, or customers' designs. We discovered that many companies have abandoned their efforts to understand exactly what products their customers want and have instead equipped them with tools to design and develop their own products. The user-centered design models should be equipped with a set of facilitating toolkits in order for customers to interact with the system. One practicable user-centered design system refers to "customer innovation" [2], [3] and is well known by industries. The conjoint analysis-based approaches [4], [5], [6] are also available for implementing user-centered design systems.

In most cases, the customers' designs are handled as a sub-process of product development. The product profiles designed by the customers are sent back to the manufacturers for analysis in order to develop new products that satisfy both the customers' wants and the manufacturers' profit requirements. In other cases, the manufacturers embed cost information in toolkits and use the cost information as constraints when customers use the toolkits to design their products. This "customers interactively design with manufacturers" strategy is applicable to B2C and B2B e-commerce, but might not be able to survive personalization requirements and a quickly changing environment.

A better method of meeting customers' needs is allowing them to design product profiles completely by themselves and making the products accordingly. A new C2C e-commerce framework can be created if C2C market providers could equip C2C buyers with a set of design toolkits that allow them to transfer their wants into product profiles while allowing numerous individual sellers on the web to negotiate with the buyers for an acceptable price.

2.2 Manipulated Fitness Assignment of Interactive Genetic Algorithms

Genetic algorithms (GA) need a fitness measurement for each individual chromosome in every generation in order to guide the evolution. Therefore, we must carefully design a pertinent fitness function and use it to generate the fitness. We can apply IGA to solve problems that have difficulties with designing a fitness function. In this case, the user serves as a fitness function and assigns fitness to the chromosomes. Designing an IGA fitness assignment method is an interesting future research topic. Conventional fitness assignment strategy asks users to assign fitness to every individual chromosome of the population, known as the rating all strategy [7]. Other assignment strategies, such as the bias strategy, the picking some up strategy [8], and the distance-based bias strategy [9] have also addressed the need to improve the performance of IGA.

Assigning fitness during evolutions places a heavy load on IGA users during the interaction activities. This results in the primary fatigue problem in IGA. Although many IGA-based systems have been proposed for real world cases, the fatigue problem still needs to be solved.

The select operations in GA are divided into three parts:

1. Calculating the fitness of each chromosome according to a pre-designed fitness function.
2. Deciding which chromosomes should be selected into the mating pool according to their fitness.
3. Selecting pairs of chromosomes for crossover and/or selecting a chromosome for mutation.

In GA, a computer executes all three parts. In conventional IGA, a human does the first part, while a computer executes the other two parts. Are there any strategies, however, to manipulate the other two parts of the select operations to improve IGA performance? In this study, we propose a manipulated selection interactive genetic algorithm (SIGA), where humans do the first and second parts of the select operations, while a computer executes the third part.

3 Research Framework and Models

3.1 The Customizable C2C E-Commerce Framework

In the traditional "manufacturers/stores/Internet stores-buyers" framework, three implicit assumptions are recognized: (1) Manufacturers well understand customers' needs or wants, (2) buyers well know their preferences, and (3) the products in shopping malls or Internet stores are enough to satisfy all buyers' wants. The facts show that customers at traditional shopping malls usually spend a large amount of time looking for their wants, but few of these wants are ultimately satisfied. Therefore, a customizable C2C e-commerce framework (*Fig. 1*) is proposed to give unsatisfied customers another option to fulfill their needs or wants.

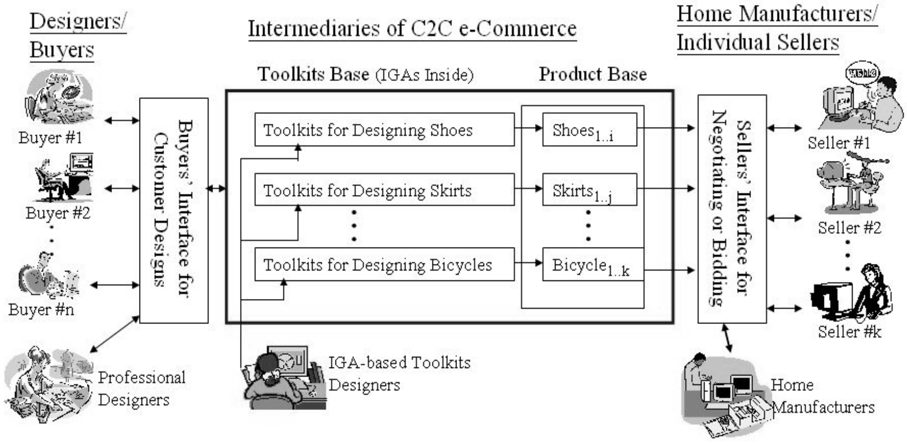

Designers/ Buyers
Intermediaries of C2C e-Commerce
Home Manufacturers/ Individual Sellers
Toolkits Base (IGAs Inside)
Product Base
Buyer #1
Buyer #2
Buyer #n
Buyers' Interface for Customer Designs
Toolkits for Designing Shoes
Toolkits for Designing Skirts
Toolkits for Designing Bicycles
Shoes1..i
Skirts1..j
Bicycle1..k
Sellers' Interface for Negotiating or Bidding
Seller #1
Seller #2
Seller #k
Professional Designers
IGA-based Toolkits Designers
Home Manufacturers

Fig. 1. The customizable C2C e-commerce framework

The product profile in the framework is defined as follows: A product P is composed of n attributes ($P = (a_1, a_2, \ldots, a_n)$). If attribute a_i has l_i attribute levels, then the size of the product space, denoted by S, is equal to $l_1 \times l_2 \ldots \times l_n$. The buyers can design their ideal products' profiles within S.

The framework includes: (1) many individual buyers, including professional designers, (2) an intermediary (like e-Bay) or a market provider, and (3) many individual sellers and some home manufacturers. The intermediary is responsible for providing a set of customers' design toolkits. The customers' design toolkits allow customers (buyers) to design many kinds of products by themselves. The intermediary is also responsible for providing a product list sub-system and a negotiating sub-system. The product list sub-system collects and catalogs the products designed by the buyers and announces the product lists on web pages; the negotiating sub-system provides potential sellers communication channels with which to advertise the designed products and to negotiate prices and delivery dates with buyers (or to bid for target products).

The design toolkits contain an efficient IGA and it is expected that somewhere in the world, there will exist at least one seller who has the producing competence to fulfill a certain demand. We also can expect that at initial stages of the customizable C2C framework, most of the demand might be met by hand, and later with computer-aided manufacturer systems or other systems.

3.2 The Benchmark Model: Over-sampling-based IGA (OIGA)

To improve IGA performance in customer designs, Hsu and Huang [10] proposed a customer values-based IGA to solve the fatigue problem. To ensure that the search space is a complete union set and is suitable for different customers, they followed Keeney's value-focused thinking [11] to build a sufficiently large product space. The results of a case study show the model is significantly helpful for improving IGA performance in customer designs.

After that, Hung and Hsu [12] proposed an over-sampling-based IGA (OIGA), which not only followed Keeney's value-focused thinking approach, but also allowed IGA users to be involved in generating the first population. Since the over-sampling strategy can ensure that a suitable proportion is prepared in the first generation, the case study results show that the strategy performs as expected.

In this study, the OIGA model is used as a benchmark model for comparison with the proposed model. The OIGA model procedure is shown below.

```
OIGA( )
{
Build a search space using Keeney's value-focused
thinking approach.
Generate a first generation of chromosomes with over-
sampling strategy.
Do
{
Phenotypes of the chromosomes are shown to the user.
The user assigns fitness to each chromosome.
IGA selects chromosomes into the mating pool according
to the fitness.
IGA generates new chromosomes of the next generation by
applying crossover and/or mutation.
} Continue until the user has found a satisfactory
chromosome or has reached other ending conditions.
```

3.3 The Proposed Model: Entrusting Select Operations of IGA to Customers

The IGA fitness assignment processes are time-consuming and tedious, but any inaccurate rating causes IGA to select improper chromosomes into the mating pool, resulting in user fatigue.

The actual purpose of the fitness function in GA is to calculate fitness values of the chromosomes. Some fit chromosomes are then selected into a mating pool accordingly. The real key is determining which chromosomes should be selected into the mating pool.

The fitness function and the fitness value are both methods of selecting chromosomes into the mating pool. Allowing IGA users to directly select fit chromosomes into the mating pool, however, is a very natural and effortless method. In other words, select operations of IGA do not necessarily require fitness functions and fitness value. We should allow IGA users to bypass the fitness assignment processes and let them directly select fit chromosomes into the pool according to their hidden fitness functions. Therefore, we propose a model that allows users to directly select chromosomes from the n^{th} generation into the mating pool to evolve the next generation of chromosomes. The proposed model refers to SIGA, which introduces human capabilities slightly more than simple IGA. The SIGA procedure is shown below.

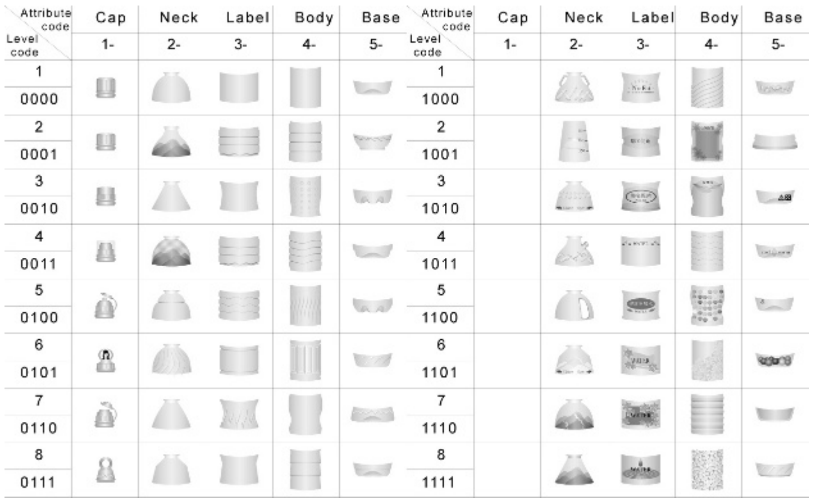
```
SIGA( )
{
Build a search space using Keeney's value-focused
thinking approach.
Generate a first generation of chromosomes with over-
sampling strategy.
Do
{
Phenotypes of the chromosomes are shown to the user.
The user selects k (0<= k <= n, n = population size)
chromosomes into the mating pool.
IGA randomly generates (n-k) chromosomes into the pool.
IGA generates new chromosomes of the next generation by
applying crossover and/or mutation.
} Continue until the user has found a satisfactory
chromosome or has reached other ending conditions.
```

4 A Case Study: Designing Water Bottles with SIGA and OIGA

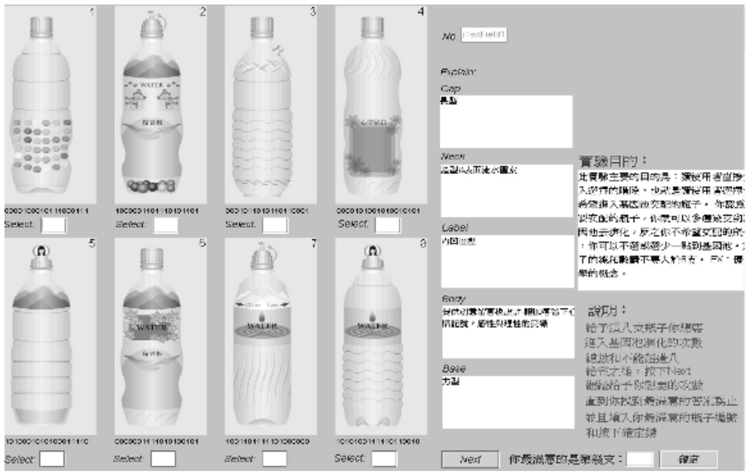
4.1 The Chromosome Structure of Mineral Water Bottles

Mineral water bottle design was used as a case study to verify the performance of the proposed model. A mineral water bottle has five attributes: cap, neck, label, body, and base; the cap itself has 8 attribute levels, while the other attributes take on 16 attribute levels. The size of the solution space (bottles) is 2^{19} (= $2^3 \times 2^4 \times 2^4 \times 2^4 \times 2^4$ = *524,288*). The chromosome structure, genotype, and the phenotype of the encoded attribute levels are created by following the value-focused thinking approach. The results are shown in *Table 1*.

Table 1. The chromosome structure, genotype, and phenotype of the mineral water bottles



Attribute code / Level code	Cap 1-	Neck 2-	Label 3-	Body 4-	Base 5-	Attribute code / Level code	Cap 1-	Neck 2-	Label 3-	Body 4-	Base 5-
1 / 0000						1 / 1000					
2 / 0001						2 / 1001					
3 / 0010						3 / 1010					
4 / 0011						4 / 1011					
5 / 0100						5 / 1100					
6 / 0101						6 / 1101					
7 / 0110						7 / 1110					
8 / 0111						8 / 1111					



說明：
給予這八支瓶子你想要
進入基因池演化的次數
總數和不能超過八
給完之後，按下Next
繼續給予你喜愛的次數
直到你找到最滿意的答案為止
並且填入你最滿意的瓶子編號
和按下確定鍵

Fig. 2. The interface of the SIGA system

4.2 Experimental Designs

We invited anyone who was interested in customers' designs to take part in this water bottle design experiment. Each subject was asked to design his or her preferred bottle from *524,288* candidate bottles using OIGA and SIGA systems separately. The experiment randomly provided OIGA-based and SIGA-based systems. *Fig. 2* shows the SIGA-based system interface. We recorded *65* valid records at the end of the experiment.

During the experiments, we recorded the number of generations every subject actually used in both systems, and set those numbers as the efficiency indexes ($E_{i\text{-}OIGA,\ i=1...65}$ and $E_{i\text{-}SIGA,\ i=1...65}$). At the end of the tests, both on OIGA and SIGA, subjects were asked to choose one satisfactory bottle from the chromosomes of the last generation, and rated the bottle on a *100*-point scale for a satisfaction score. We used these scores as the effective indexes ($F_{i\text{-}OIGA,\ i=1...65}$ and $F_{i\text{-}SIGA,\ i=1...65}$).

In the experiments, the population size was set to *8*, one-point crossover was used, one elitist bottle in each generation was preserved in the next generation, the crossover rate and mutation rate were *0.8* and *0.01* respectively, and the over-sampling rate was *0.7* [10], [12].

5 Experimental Results

The results of the efficiency experiments are shown in *Fig. 3*. They show that no matter which system the subjects used, most of them completed their designs within

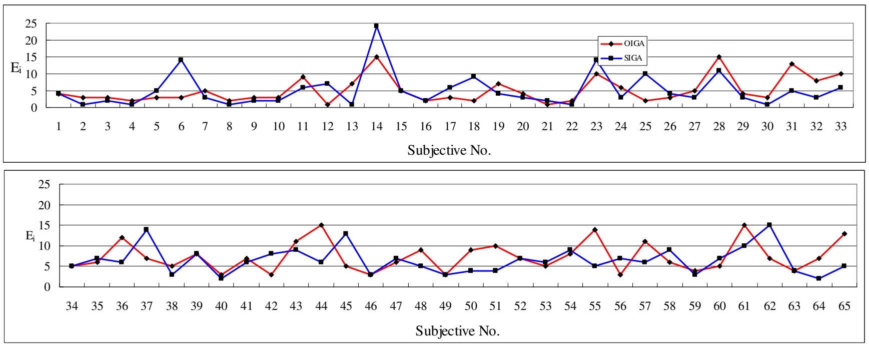


Fig. 3. Results of the efficiency test

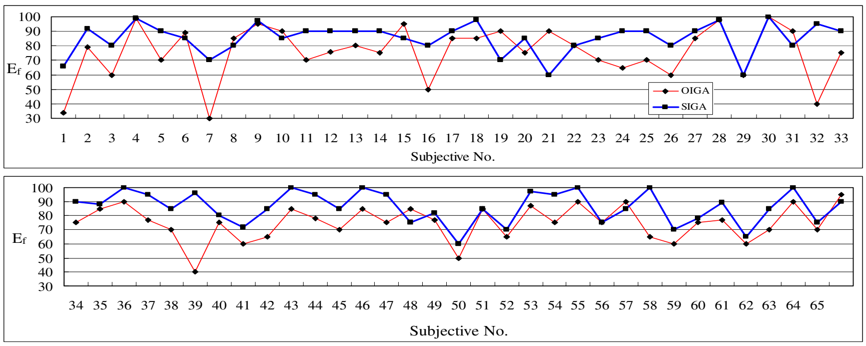
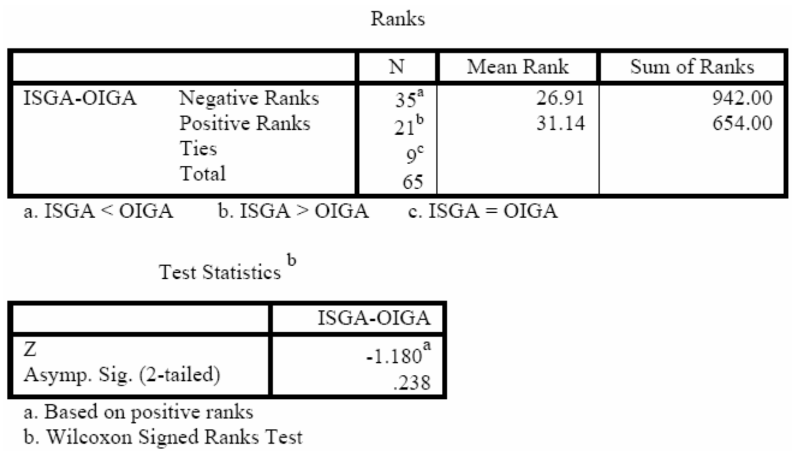


Fig. 4. Results of the effective test

10 generations. The results of the Wilcoxon signed-ranked test on the system's efficiency index are shown in *Table 2*. The results in *Table 2* indicate that the efficiency index of OIGA is not significantly different from SIGA (*p =0.238*).

Fig. 4 and *Table32* show experimental results from the effective test and the results of the Wilcoxon signed-ranked test on the effective index. The results indicate that SIGA models are more effective than OIGA, and the SIGA's effective index is significantly different from OIGA (*p =0.000*).

Table 2. Results of the Wilcoxon Signed-Ranked Test on efficiency index



Ranks

		N	Mean Rank	Sum of Ranks
ISGA-OIGA	Negative Ranks	35[a]	26.91	942.00
	Positive Ranks	21[b]	31.14	654.00
	Ties	9[c]		
	Total	65		

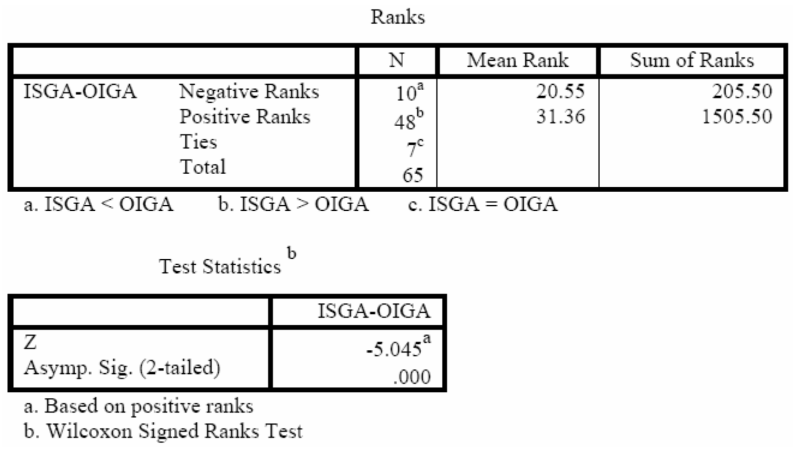
a. ISGA < OIGA b. ISGA > OIGA c. ISGA = OIGA

Test Statistics[b]

	ISGA-OIGA
Z	-1.180[a]
Asymp. Sig. (2-tailed)	.238

a. Based on positive ranks
b. Wilcoxon Signed Ranks Test

Table 3. Results of the Wilcoxon Signed-Ranked Test on effective index



Ranks

		N	Mean Rank	Sum of Ranks
ISGA-OIGA	Negative Ranks	10[a]	20.55	205.50
	Positive Ranks	48[b]	31.36	1505.50
	Ties	7[c]		
	Total	65		

a. ISGA < OIGA b. ISGA > OIGA c. ISGA = OIGA

Test Statistics[b]

	ISGA-OIGA
Z	-5.045[a]
Asymp. Sig. (2-tailed)	.000

a. Based on positive ranks
b. Wilcoxon Signed Ranks Test

6 Concluding Remarks

In this paper, we propose a customizable C2C framework to fully utilize interactive genetic algorithms and to discover the potential capabilities of IGAs in customer designs. The traditional interactive IGA methods are unnatural, especially when IGA is applied to customers' designs. In this study, we found that allowing IGA users to directly select chromosomes into the mating pool is not only a natural way to implement the select operations of IGA, but also more effective than the traditional method.

Freeing users from fatigue is the most important challenge in IGA. The OIGA (Over-sampling IGA) model has shown its effectiveness at decreasing fatigue. We conducted a case study and used the OIGA model as a benchmark. The results show that the proposed SIGA model is significantly more effective than the OIGA model. This means that users who design their product profiles with the benchmark system will achieve the same satisfaction level as with the proposed system, but will use more IGA generations. These extra generations will cause user fatigue. In other words, the evidence shows that the proposed model performs better than the benchmark system in terms of preventing fatigue.

To create an effective, applicable, and fatigue-free IGA, the interaction between humans and computer systems in IGA must be extended. One possible extension is allowing humans to directly interact with the genome, the genetic operators, or the evolution processes; another possible extension is allowing humans to cooperate with computational intelligence by guiding human-assigned fitness accurately or directing IGA operations in the chromosome effectively.

Acknowledgement

Financial supports from Taiwan National Scientific Council (NSC 94-2416-H-156-001-) are gratefully acknowledged.

References

1. Takagi, H: Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation, in Proceedings of IEEE, Vol. 89, No. 9 (2001) 1275-1296.
2. Urban, G. L. and von Hippel, E.: Lead User Analyses for the Development of New Industrial Products, Management Science Vol. 34, No. 5 (1988) 569-582.
3. Thomke, S. and von Hippel, E.: Customers as Innovators - a New Way to Create Value, Harvard Business Review, Vol. 80, April (2002) 74-81.
4. Dahan, E. and Hauser, J.R.: The Virtual Customer, Journal of Product Innovation Management, Vol. 19, No. 5 (2001) 332-354.
5. Olivier, T., Hauser, J. R. and Simester, D. I.: Polyhedral Methods for Adaptive Choice-Based Conjoint Analysis, Journal of Marketing Research, Vol. 41, No. 1 (2004) 116-131.
6. Olivier, T., Simester, D. I., Hauser, J. R., and Dahan. E.: Fast Polyhedral Adaptive Conjoint Estimation, Marketing Science, Vol. 22, No. 3 (2003) 273-303
7. Caldwell, C. and Johnston, V. S. Tracking a Criminal Suspect through 'Face-Space' with a Genetic Algorithm, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, California (1991) 416-421.
8. Nishio, K., Murakami, M., Mizutani, E. and Honda, N.: Fuzzy Fitness Assignment in an Interactive Genetic Algorithm for a Cartoon Face Search, in Sanchez, E., Shibata, T. and Zadeh, I. A.(eds.): Genetic Algorithms and Fuzzy Logic Systems - Soft Computing Perspectives, World Scientific Publishing (1997) 175-191.

9. Hsu, F. C. and Chen, J. S.: A Study on Multi Criteria Decision Making Model: Interactive Genetic Algorithms Approach, in Proceedings of the 1999 International Conference on SMC, Tokyo, Japan (1999) 634-639.
10. Hsu, F. C. and Huang, P.: Providing an appropriate search space to solve the fatigue problem in interactive evolutionary computations, New Generation Computing, Vol. 23, No.2 (2005) 114-126.
11. Keeney, R. L.: Value-Focused Thinking: A path to Creative Decision-Making, Harvard University Press, Cambridge, Massachusetts (1992).
12. Hung, M. H. and Hsu, F. C.: Accelerating Interactive Evolutionary Computation Convergence Pace by Using Over-sampling Strategy, in The Fourth IEEE International Workshop on Soft Computing as Trans-disciplinary Science and Technology, May 25-27, Muroran, Japan (2005).

Evaluation of Sequential, Multi-objective, and Parallel Interactive Genetic Algorithms for Multi-objective Floor Plan Optimisation

Alexandra Melike Brintrup[1], Hideyuki Takagi[2], and Jeremy Ramsden[1]

[1] School of Industrial and Manufacturing Science, Cranfield University,
Bedfordshire, MK43 0AL UK
Tel.+ 44 1234 750111 ext.2281, Fax + 44 1234 751346
{A.Brintrup, J.Ramsden}@cranfield.ac.uk
[2] Faculty of Design, Kyushu University,
4-9-1, Shiobaru, Minami-ku, Fukuoka, 815-8540 Japan
Tel. & Fax: +81 92-553-4555
takagi@design.kyushu-u.ac.jp

Abstract. We propose a sequential IGA, multi-objective IGA and parallel interactive genetic algorithm (IGA), and evaluate them with a multi-objective floor planning task through both simulation and real IGA users. Combining human evaluation with an optimization system for engineering design enables us to embed domain specific knowledge which is frequently hard to describe, subjective criteria and preferences in engineering design. We introduce IGA technique to extend previous approaches with sequential single objective GA and multi-objective GA. We also introduce parallel IGA newly. Experimental results show that (1) the multi-objective IGA and the parallel IGA clearly provide better results than the sequential IGA, and (2) the multi-objective IGA provides more diverse results and faster convergence for a floor planning task although the parallel IGA provides better fitness convergence.

1 Introduction

Interactive Evolutionary Computation (IEC) is an EC that optimizes a target system based on human subjective evaluation, and the human plays the role of a fitness function of conventional EC [1]. When it is applied to fields that include a degree of subjectivity, such as engineering design, arts creation, music composition, or architecture, interaction with a human evaluator helps the EC to generate solutions that incorporate his/her expertise~~,~~ or intuition without its explicit description into the optimisation platform. Interaction between an IEC user and EC can proceed in many ways depending on task domain. For instance, the user may participate in choosing elite designs for survival, modify an individual and reinsert it into the population of designs, freeze parts of the design with the intention of reducing the search space dimensionality besides human fitness evaluation in normal IEC. Therefore, Parmee redefined IEC broadly as the system optimisation based on human-machine interaction [2].

Multi-objective design optimisation is defined as the problem of finding a vector of decision variables that optimizes a vector function whose elements represent multiple objective functions. The philosophy of multi-objective optimisation comes from the need to achieve compromise decision making in a problem of many conflicting objectives. In such an environment, the ideal platform would enable us to gather a diverse set of solutions, each with their own offering of different levels of objective satisfaction, so that a choice of solution or solutions can easily be reached.

Our previous survey [3] showed that many so-thought quantitatively dominated problems, such as engineering design or system design, had (1) multiple conflicting objectives and (2) subjective objectives among them. Subjective objectives may act as conflicting or complimentary to quantitative objectives and are unpredictable due to their nature.

Recently, the usage of EC techniques in multi-objective optimisation has become particularly popular. A multi-objective optimisation problem modelled by EC is termed as Evolutionary Multiple-objective Optimisation (EMOO). The particular suitability of EC to multi-objective optimisation tasks and their subjectivity aspect had made EMOO incorporate IEC to gather subjective opinions of the designer [3]. Some researchers followed similar ideas such as gathering objective preferences from the designer [4], or asking the designer to pick favourable search area [5], and our interactive multi-objective design optimization framework [6] used subjective opinions of the designer as an additional objective function with promising results.

In this paper, we use genetic algorithm (GA) as one of EC techniques and propose a new algorithm developed for this framework: the parallel interactive genetic algorithm (IGA) that optimises each design objective in separate population islands before merging the solutions. The results of the parallel IGA are compared with previously proposed two algorithms of the framework, the sequential IGA and the multi-objective IGA that is based on the non-dominated sorting genetic algorithm of Deb [7]. We propose three IGA's in section 2, explain experimental evaluation task and condition in section 3, discuss the experiments in sections 4 and 5, and conclude on this comparison of three IGA's with remained future work.

2 Platforms Developed for Interactive Multi-objective Optimisation

2.1 Sequential IGA

The sequential IGA acts as a single objective optimisation platform, taking turns to optimize the quantitative and qualitative objectives independently.

A single population is evolved by a sequentially switched fitness function. Initially a subjective run is performed with a conventional IGA, meaning that, the user needs to evaluate all the individuals of the current population displayed by one design at a time, by giving a qualitative rating between 0 and 9, 0 being the best design and 9 worst. Users are allowed to give the same rating to more than one design. The subjective optimization is run by the algorithm by solely taking into account the user given rating, which, to the algorithm, acts as a blackbox fitness function. A subjective generation is then followed by a given number of quantitative optimisation runs where fitness is evaluated by a regular fitness function.

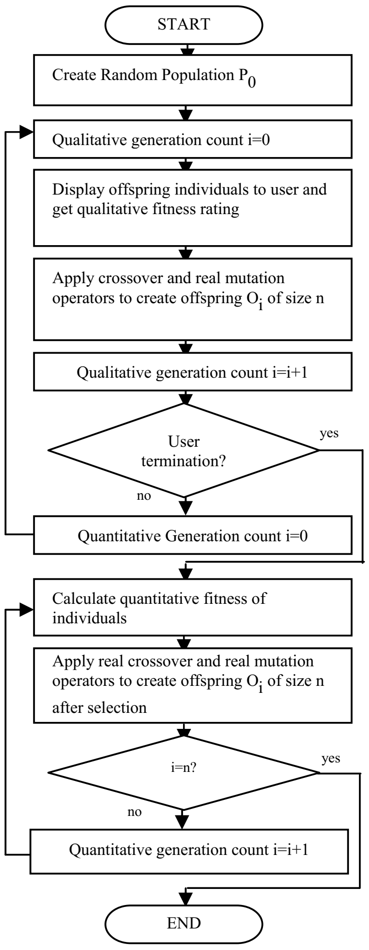

START
Create Random Population P0
Qualitative generation count i=0
Display offspring individuals to user and get qualitative fitness rating
Apply crossover and real mutation operators to create offspring Oi of size n
Qualitative generation count i=i+1
User termination?
yes
no
Quantitative Generation count i=0
Calculate quantitative fitness of individuals
Apply real crossover and real mutation operators to create offspring Oi of size n after selection
i=n?
yes
no
Quantitative generation count i=i+1
END

Fig. 1. Sequential IGA design optimization

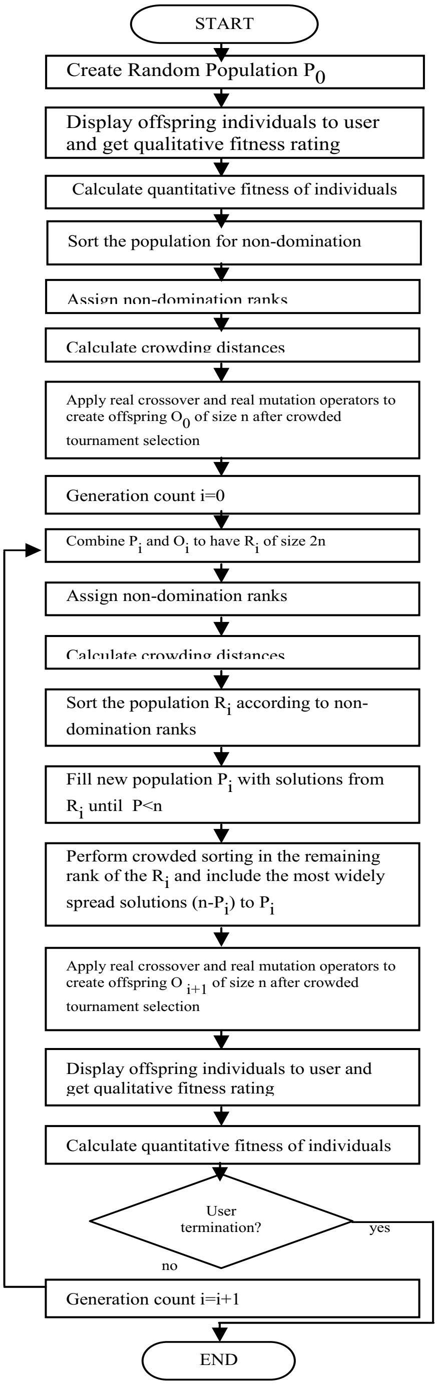

START
Create Random Population P0
Display offspring individuals to user and get qualitative fitness rating
Calculate quantitative fitness of individuals
Sort the population for non-domination
Assign non-domination ranks
Calculate crowding distances
Apply real crossover and real mutation operators to create offspring O0 of size n after crowded tournament selection
Generation count i=0
Combine Pi and Oi to have Ri of size 2n
Assign non-domination ranks
Calculate crowding distances
Sort the population Ri according to non-domination ranks
Fill new population Pi with solutions from Ri until P<n
Perform crowded sorting in the remaining rank of the Ri and include the most widely spread solutions (n-Pi) to Pi
Apply real crossover and real mutation operators to create offspring O i+1 of size n after crowded tournament selection
Display offspring individuals to user and get qualitative fitness rating
Calculate quantitative fitness of individuals
User termination?
yes
no
Generation count i=i+1
END

Fig. 2. Multi-objective IGA design optimization

This method treats the subjective and quantitative features of the design problem as separate objectives to be optimized. For instance, the individuals created from one qualitative run are fed into the following quantitative run as parent designs, which ensures the starting point of the quantitative run to be subjectively optimized designs. This is how the connection between subjective and quantitative criteria is ensured.

With this algorithm, the authors aimed to represent a typical design cycle in an engineering design firm, where the design is thrown over the wall between the marketing department, concerned with subjective aspects of the design, and the R&D department, concerned with quantitative aspects of the design. Figure 1 shows the flow of sequential IGA

2.2 Multi-objective IGA

Multi-objective IGA is based on a modified version of a popular multiple objective optimisation algorithm, the non-dominated sorting GA 2 by Deb [7].

The algorithm enhances usual non-domination based multi-objective optimisation techniques by introducing the concept of elitism and diversity. Elitism makes sure to preserve globally good solutions throughout generations. In the non-dominated sorting GA 2, elitism is achieved by combining parent and offspring populations before sorting them for non-domination. Non-domination of a solution shows how many solutions are better in all criteria from the current solution. The non-dominated sorting GA 2 also looks into obtaining as diverse solutions on the Pareto front (the solution set, observed when there are conflicting objectives, which contains solutions that are non-dominating to each other) as possible, by a performing calculation called, crowding distance. Readers may refer to Deb [7] for a detailed description of the crowded tournament operator and the overall non-dominated sorting GA 2 procedure.

We modify the Deb et al.'s GA to include interactive fitness gathering and renamed it as the *multi-objective IGA* in our study. In multi-objective IGA also a single population is optimised. The subjective objective value is gathered from the user for a single solution, whereas the quantitative fitness of a solution is assessed by the built-in fitness function. Figure 2 shows the flow of multi-objective IGA.

2.3 Parallel IGA

EC techniques are suitable for parallelization, as crossover, mutation, and evaluation can be performed independently on different individuals. It is possible to separate individuals themselves to be evolved in different processors or programs, or separate each location or program to perform different selection and recombination routines on each individual. In either case, the reason of the choice of parallelization depends very much on the problem at hand. For instance, parallelization can be a solution for computationally demanding problems, to apply selection and recombination routines to individuals, or it can simply be a way to separate populations, as in our case. In any of these cases, three main parallelization techniques are used widely [8]:

- Master-slave parallelization, where a single processor maintains control over selection and uses the other processors only for crossover, mutation and evaluation of individuals. This is useful for few processors and very large evaluation times.

- Island model, where every processor runs an independent EC, using a separate sub-population. The processors cooperate by regularly exchanging migrants (good individuals). This technique model is suitable for clustering populations.
- Diffusion model, where the individuals mate with other individuals only with the local neighbourhood. This approach is particularly suitable for massively parallel computers with a fast local intercommunication network.

The usage of parallelization on EMOO can be effective as the goal of EMOO is to find a set of good solutions rather than a single optimum. Nevertheless there exists a limited amount of literature on their usage in this field [9][10]. On the other hand, the usage of parallelization techniques in IEC has not been reported.

For the purposes of our research, we hypothesize that the usage of parallelization and the usage of interactivity together could be advantageous and natural for our problem. Since we deal with multiple conflicting objectives these could be evolved with separate populations with elite migrants exchanged between one another i.e. following the island model. The subjective objective fitness of a solution could be obtained by user interaction and used to evolve one population while the other population is evolved by a regular fitness function. It has been hypothesized that the obvious advantages of this method would be (1) quantitatively evolved population could be evolved much faster, utilizing the time by the human-computer interaction, (2) a compromise decision can be encouraged by the migration of elites between populations.

The features of the parallel IGA include parallelisation technique, immigrant selection, replacement strategy, and fitness assignment strategy to immigrants:

Parallelisation technique: the parallel IGA uses an island model and optimises n separate populations with n separate objectives with immigrants exchanged among them. In our experiments, we use $n = 2$.

Immigrant selection: the top three elite solutions are selected from each population to be migrated.

Replacement strategy: the worst three individual solutions from each population are replaced with the immigrants

Fitness assignment strategy to immigrants:

- In the population optimised using the quantitative objective, immigrants are sorted with respect to IGA rating. If any two ratings are equal, sorting is done using the calculated quantitative objective fitness. After sorting, an arbitrary quantitative fitness is assigned to the immigrants to ensure their survival. The arbitrary fitness assignment proceeds with dividing the range of fitness obtained into five main regions. Then, the mid, top, and bottom fitness levels of the top fitness range (i.e. the minimum fitness interval as the problem is modelled as a minimisation problem) is assigned to the three immigrants.
- In the population optimised using the quantitative objective, immigrants are all given the minimum, i.e. the best IGA rating. The reason is that the subjective fitness rating is a discrete value and designs taking the same rating value are allowed. However, there is very little probability that any two designs would give the same quantitative fitness evaluation. Even though two designs may differ from each other, the user might give the same rating in contrast to the quantitative fitness evaluation. The parallel IGA fitness assignment strategy therefore is represents this phenomenon.

Two parallel IGA platforms are used for testing: the pseudo parallel IGA that uses a fitness function as a pseudo IGA user and the real parallel IGA that is a normal IGA with a real human user. The user evaluation of the pseudo parallel IGA is simulated by a fitness function and no real user involvement is included. The pseudo parallel IGA is conducted to evaluate its maximum performance under the ideal condition without human fluctuation in his/her evaluation. Figure 3 shows the flow of the general parallel IGA procedure. Pseudo parallel IGA is coded with the C language whereas real parallel IGA is written with C++, using the Microsoft Foundation Class Library (MFC), Open Graphics Library (OpenGL) and Coin3D Library[1].

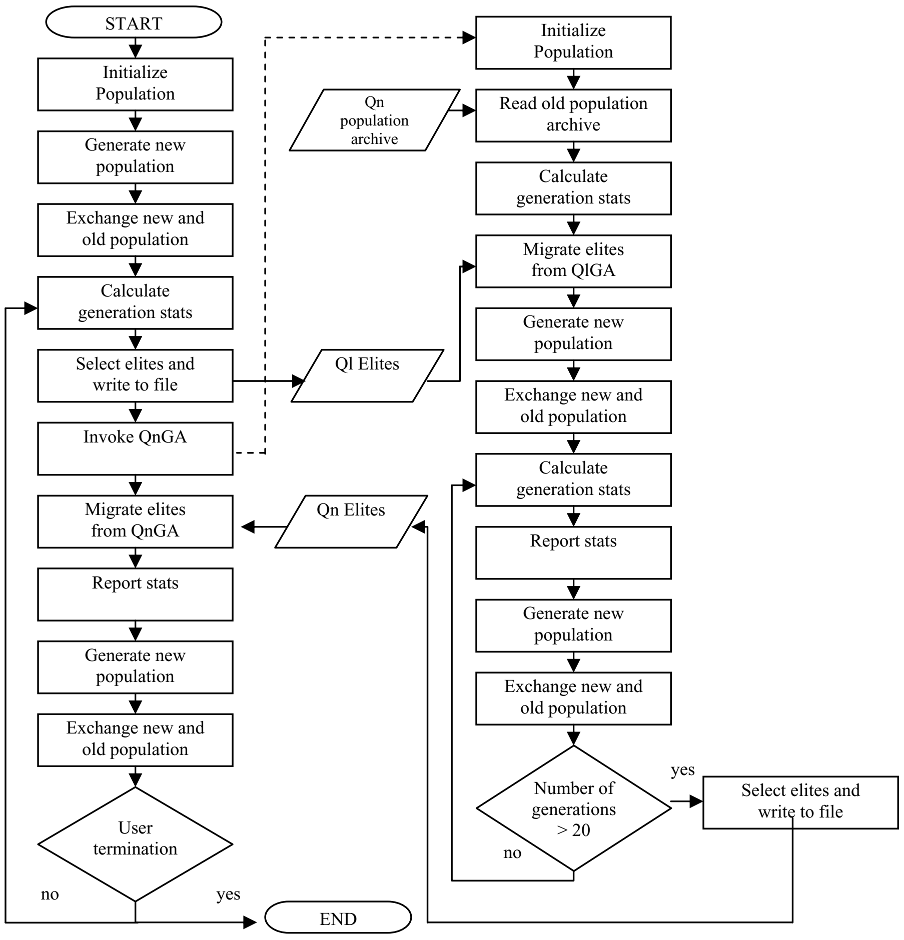

START
Initialize Population
Generate new population
Exchange new and old population
Calculate generation stats
Select elites and write to file
Invoke QnGA
Migrate elites from QnGA
Report stats
Generate new population
Exchange new and old population
User termination
no
yes
END
Initialize Population
Qn population archive
Read old population archive
Calculate generation stats
Migrate elites from QlGA
Ql Elites
Generate new population
Exchange new and old population
Calculate generation stats
Qn Elites
Report stats
Generate new population
Exchange new and old population
Number of generations > 20
yes
no
Select elites and write to file

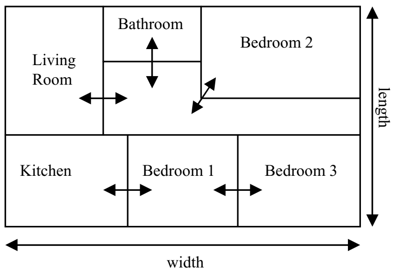
Fig. 3. Parallel IGA flow

[1] Please refer to http://msdn.microsoft.com, www.opengl.org, and www.coind3d.org for more information on these graphics libraries.

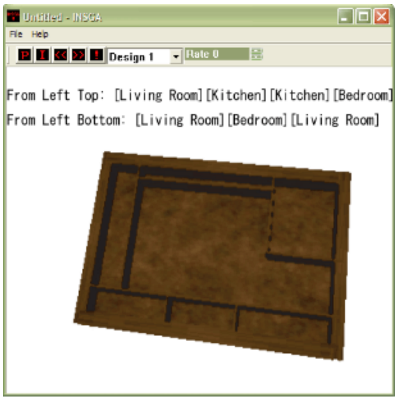
3 Experimental Evaluation with Floor Planning Design

3.1 Task Description

The house floor planning design is used as a benchmark task. The objectives are to find the width and length of each room as shown on Figure 4(a). These parameters will (1) minimize the cost of build which directly relates to the total area of the floor plan, and (2) maximize subjective user evaluation received by the interactive fitness evaluation element. This problem is an ideal candidate for testing the developed algorithms as it includes both quantitative and subjective features. Minimization of cost clearly constitutes a quantitative feature. On the other hand, the arrangement and sizes of the rooms vary from person to person and can be easily subjectively evaluated. Table 1 shows the parameters of the problem and how the overall area is



(a)



From Left Top: [Living Room][Kitchen][Kitchen][Bedroom]
From Left Bottom: [Living Room][Bedroom][Living Room]

(b)

Fig. 4. (a) Dimensional variability in the floor plan design, (b) Graphical user interface of the modelled floor plan design problem

Table 1. Floor plan design parameters

Room	Parameter	Parameter Label	Area
Living room	width	X_0	$A_1=X_0\,(2.2 - X_1)$
Kitchen	length	X_1	$A_2=2\,X_1\,X_2$
Kitchen	width	X_2	
Bedroom 1	width	X_3	$A_3=X_3\,X_4$
Bedroom 1	length	X_4	
Bathroom	length	X_5	$A_4=2*[3.6 - (X_0+X_6)]*X_5$
Bedroom 2	width	X_6	$A_5=X_6\,X_7$
Bedroom 2	length	X_7	
Bedroom 3	-	-	$A_6=X_1*[3.6 - (X_2+X_3)]$
Hall	-	-	$A_7=[3.6 - (X_0+X_6)]*[2.2 - (X_1+X_5)]+X_6*[2.2 - (X_1+X_7)]$

deduced from these parameters. Detailed description of the problem can be found in [3]. Figure 4(b) is an example of the user interface used for our experiments.

The users are asked to evaluate the design by paying special attention to the sizes of the bathroom and kitchen. The greater the sizes of these rooms, the greater the user satisfaction should be. The reasons for this included: (1) ensuring the subjective and quantitative objectives to be conflicting such that a pareto front could be reached and analysed, (2) ensuring consistency between pseudo user evaluation as obtained from the pseudo parallel IGA and the real user evaluation in the subjective components of the rest of algorithms developed. Table 2 shows the fitness evaluation method for subjective and quantitative objectives in the relevant components of the three different algorithms.

Table 2. Fitness evaluation in quantitative and subjective components of the parallel IGA, multi-objective IGA, and sequential IGA

	Description	*Component*
Real human evaluation	A 10-scale subjective rating is taken from the user.	Real parallel IGA subjective island Multi-objective IGA subjective component Sequential IGA subjective component
$\sum_{i=1}^{7} A_i$	This function minimizes the total area of the rooms thus reduces cost which is directly proportional to the total area of the floor plan.	Pseudo parallel IGA quantitative island Real parallel IGA quantitative island Multi-objective IGA quantitative component Sequential IGA quantitative component
$\frac{1}{A_2 + A_4}$	This function simulates a user whose requirement is bigger bathroom and kitchen areas.	Pseudo parallel IGA subjective island

3.2 Test Parameters

Real GA coding, tournament selection, a mutation rate of 0.01, one-point simulated binary crossover with a rate of 0.9, distribution indexes of 20 for Simulated Binary Crossover and 10 for real mutation are used in both parallel IGA islands, sequential IGA and multi-objective IGA.

GA properties such as recombination and selection operators, probabilities for selection, recombination and mutation, and population sizes are kept constant throughout to allow an accurate comparison of the three approaches.

Experimental subjects are 2 females and 3 males whose ages range in 22-26. The subjects' expertise included three product designers, one engineer, and one architectural engineer.

Users continued to run the program until subjective generation 5 was reached. Each user conducts one test for each of parallel IGA, sequential IGA, and multi-objective IGA.

For sequential IGA, with each user, 6 runs are performed, 3 of which are subjective and 3 of which are quantitative runs. For these, each subjective run consisted of 5 generations, while each quantitative run consists of 10 generations.

For multi-objective IGA, with each user, one run of 5 generations are performed.

For parallel IGA, the number of generations and the population size for each island is given in Table 3.

Table 3. Parallel IGA parameters for quantitative and subjective population islands

	Population size	Number of generations at each run	Total number of generations	Total number of user evaluations
Parallel IGA - Quantitatively optimised population island	50	10	50	0
Parallel IGA - Subjectively optimised population island	12	1	5	60
Multi-objective IGA	12	1	5	60
Sequential IGA - Quantitative optimisation run	12	10	50	0
Sequential IGA - Subjective optimisation run	12	1	5	60

4 Results

Sequential IGA, multi-objective, and IGA Parallel IGA are compared over 55, 5, and 55 generations, respectively, in terms of overall average subjective fitness, overall average quantitative fitness, average subjective fitness of last generation, and average quantitative fitness of last generation. The Wilcoxon signed rank test, which is a nonparametric pair observation test, is used to compare the results from these three algorithms.

Figure 5 shows the average fitness values obtained at each generation of each run. In the sequential IGA the qualitative average fitness showed a worsening trend in the five runs pursued, while the quantitative results showed a result that was smoothly improving. In the multi-objective IGA, parallel values were obtained as opposed to sequential in sequential IGA. The qualitative and quantitative fitness averages both showed an improving trend and the convergence to Pareto front was observed. As the number of generations increase the solutions are minimized with respect to both criteria. For the parallel IGA, the Wilcoxon signed rank test indicates that:

- The pseudo parallel IGA is significantly advantageous over both sequential IGA and multi-objective IGA in terms of overall average subjective fitness, overall average quantitative fitness, average subjective fitness of last generation, and average quantitative fitness of last generation, with a risk of 0.05.
- The real parallel IGA is significantly advantageous over the multi-objective IGA in terms of overall average subjective fitness, overall average quantitative fitness, and average quantitative fitness of last generation, with a risk of 0.05. The significance of average subjective fitness of last generation could not be concluded with the number of tests performed.
- The real parallel IGA is significantly advantageous over the sequential IGA in terms of overall average subjective fitness, overall average subjective fitness, and average quantitative fitness of last generation, with a risk of 0.05. The significance of average quantitative fitness of last generation could not be concluded with the number of tests performed.

- We could conclude that the parallel IGA then provides significantly better results in terms of multi-objective IGA and sequential IGA under our experimental conditions although no significance was concluded for average last generation subjective and quantitative fitnesses. Figure 5 shows the change in fitness in subjective and quantitative fitness values in the three algorithms. The following section discusses these findings.

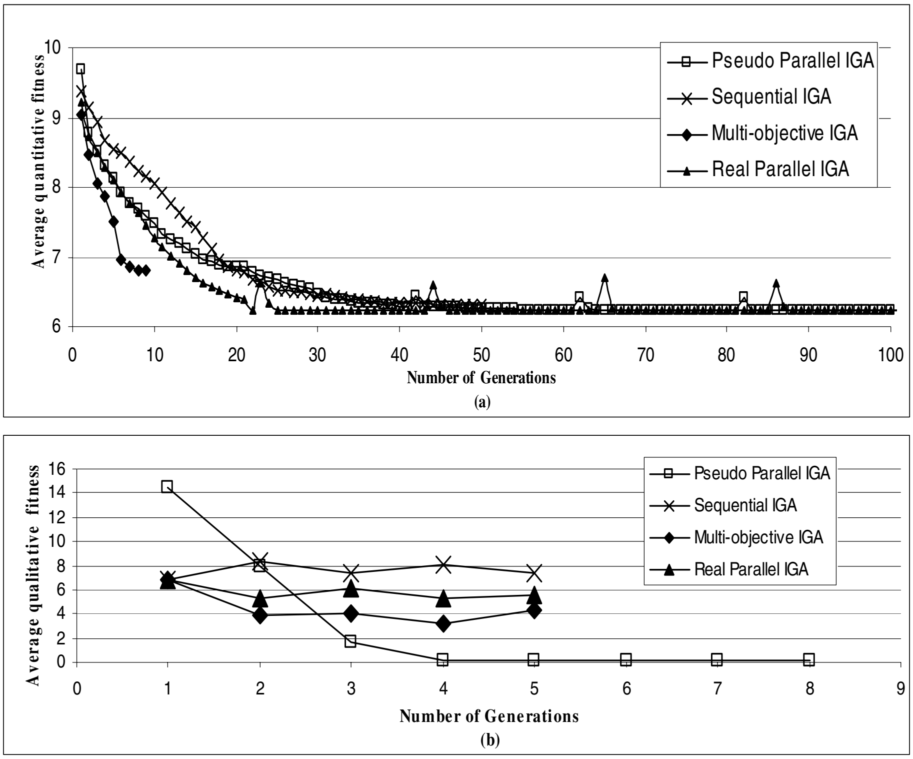


Fig. 5. Change of average (a) quantitative and (b) qualitative fitness in sequential IGA, multi-objective IGA, pseudo IGA, and real parallel IGA

5 Discussions

In multi-objective problems, it is important that a set of good solutions that are diverse from each other are obtained, so that compromise decision making can be implemented. This section discusses the results in terms of fitness convergence and diversity of results, after laying out the main sources of error.

(A) Sources of Error
Pseudo parallel IGA showed significantly better fitness convergence in both subjective and quantitative objective values than the multi-objective IGA and the

sequential IGA. However, the multi-objective IGA and the sequential IGA involved real human evaluation whereas parallel IGA did not. In order to minimise the tendency for this error, during user evaluation in the multi-objective IGA and the sequential IGA, users were asked to evaluate designs by observing the sizes of the kitchen and bathroom, hence to resemble the fitness function in parallel IGA. However, noise due to human inconsistency is inevitable and should be taken into account in the assessment of results.

Another point to bear in mind while assessing these results is the number of generations pursued with each algorithm. The multi-objective IGA pursues one quantitative and one subjective generation together at a time since the evolution is simultaneous. As the user must be involved in every generation and his/her fatigue must be considered, only 5 quantitative generations could be evolved as opposed to 55 in the sequential IGA and the parallel IGA. We can observe that even with only five generations, the performance of multi-objective IGA in both objective values was comparable to that of the parallel IGA and the sequential IGA.

(B) Fitness Convergence

The multi-objective IGA and parallel IGA reached satisfactory designs in a shorter time of 4 to 5 generations, whereas the sequential IGA failed to reach an equally satisfactory design in the qualitative and quantitative objective spaces in a total of 45 generations. Although the final fitness scores of the quantitative objectives were better in the sequential IGA and the parallel IGA than the multi-objective IGA since the subjective objective was of equal importance to the design overall, the sequential IGA performed significantly worse than other two IGA's. Rather than reaching a *compromise* decision of equal qualitative and quantitative factors, the sequential IGA competed the two objectives against each other; resetting and trying to recover from the affects of the opposite objective each time. On the other hand, no significant difference was found between averages of parallel IGA and multi-objective IGA at the final generation although the overall fitness average of parallel IGA is better than that of multi-objective IGA.

(C) Diversity

After the second quantitative run of the sequential IGA, the quantitative objective has taken over providing designs with little or no difference for the qualitative evaluation by the user. The users reported difficulty in distinguishing the designs, even though minor differences still existed, but were difficult to be visualized. This led the users to give similar ratings to designs, and it became difficult for the algorithm to diversify designs. On the other hand, the diversity preservation mechanism with the help of the crowding distance calculation in the multi-objective IGA has provided results that were visually distinct from each other. Although no quantification of diversity was performed, the diversity level in decreasing order was with multi-objective IGA, parallel IGA, and sequential IGA.

6 Conclusions

This paper compared three IGA algorithms, namely parallel IGA, multi-objective IGA and sequential IGA that are developed to optimise conflicting subjective and quantitative multi-objectives. The algorithms are evaluated with the floor planning design problem.

The major advantage of parallel IGA is its flexibility to accommodate more than one population. The population size of multi-objective IGA has to be constant as multiple objectives are dealt with simultaneously on each design, while that of the parallel IGA is adjustable according to human limitations in the subjectively evolved designs and according to the potential of the computer in the quantitatively evolved designs. So the time spent by human during design evaluation is utilised and as more number of generations can be evolved in the quantitative objective, we can get a better fitness in this objective space. As there exist different populations, the parallel IGA does not engage in a fight between the two contradictory objectives, as was the case of sequential IGA. Thus we can conclude that both the multi-objective IGA and the parallel IGA are significantly better than the sequential IGA and that sequential optimisation did not give satisfactory results in dealing with multiple objectives in conflict. The parallel IGA is observed to be satisfactory for incorporating multiple criteria principles even though in itself the algorithm is not a multi-objective optimisation algorithm. In dealing with subjective and quantitative conflicting design objectives, both multi-objective IGA and the parallel IGA seem to be promising approaches.

Although the quantitative objective remains implicit to the multi-objective IGA, the parallel IGA displays the user the designs that emigrated from the quantitative objective island to the subjective objective island. Therefore, in addition to the above advantage, real parallel IGA can help promote innovative decision making by making the user observe computer generated results and reconsider the evaluation given.

Future work will focus on investigating parallel IGA's performance with other multi-objective optimisation algorithms and with different benchmark problems. Additionally, a pair-wise preference experiment is to be undertaken for the final populations achieved from the two algorithms.

Acknowledgements

This work was supported in part by 2005 Summer Program of Japan Society for the Promotion of Science.

References

1. Takagi, H.: Interactive evolutionary computation: fusion of the capacities of EC optimization and human evaluation, Proceedings of the IEEE, 89(9) (2001) 1275-1296.
2. Parmee I. C.: Poor-definition, uncertainty and human factors - a case for interactive evolutionary problem reformulation?, Genetic Evolutionary Computing Conference (GECCO2003), 3rd IEC Workshop, Chicago, USA (July 2003)
3. Brintrup A., Ramsden J., and Tiwari A.: Integrated qualitativeness in design by multi-objective optimization and interactive evolutionary computation, IEEE Congress on Evolutionary Computation (CEC2005), Edinburgh, UK (September 2005), 2154-2160
4. Parmee I. C., Cvetkovic D., Watson A., and Bonham C.: Multi-objective satisfaction within an interactive evolutionary design environment, Journal of Evolutionary Computation, MIT Press, 8(2) (2000) 197 – 222.

5. Kamalian R., Takagi H., and Agogino A.: Optimized design of MEMS by evolutionary multi-objective optimization with interactive evolutionary computation, Genetic and Evolutionary Computation Conference (GECCO2004), Seattle, USA (2004) 1030-1041.
6. Brintrup A., Tiwari A., and Gao J.: Handling qualitativeness in evolutionary multiple objective engineering design optimization, Enformatica, 1 (2004) 236-240.
7. Deb K., Agrawal S., Pratap A., and Meyarivan T.: A fast elitist non dominated sorting genetic algorithm for multi objective optimization: NSGA-2, Proc. Parallel Problem Solving from Nature (PPSN2000), Paris, France (2000) 858-862.
8. Cantu-Paz E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell, USA (2000).
9. Van Veldhuizen D. A., Zydallis J. B., and Lamont G. B.: Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 7(2) (2003) 144–173.
10. Hiroyasu T., Miki M., and Watanabe S.: The new model of parallel genetic algorithm in multiobjective optimization problems -divided range multi-objective genetic algorithms. IEEE Congress on Evolutionary Computation (CEC2000), La Jolla, CA, USA (July 2000) 333–340.

Supervised Genetic Search for Parameter Selection in Painterly Rendering

John P. Collomosse

Department of Computer Science, University of Bath, Bath, UK
jpc@cs.bath.ac.uk

Abstract. This paper investigates the feasibility of evolutionary search techniques as a mechanism for interactively exploring the design space of 2D painterly renderings. Although a growing body of painterly rendering literature exists, the large number of low-level configurable parameters that feature in contemporary algorithms can be counter-intuitive for non-expert users to set. In this paper we first describe a multi-resolution painting algorithm capable of transforming photographs into paintings at interactive speeds. We then present a supervised evolutionary search process in which the user scores paintings on their aesthetics to guide the specification of their desired painterly rendering. Using our system, non-expert users are able to produce their desired aesthetic in approximately 20 mouse clicks — around half an order of magnitude faster than manual specification of individual rendering parameters by trial and error.

1 Introduction

Techniques for processing photographs into artwork have received considerable attention in recent years and comprise a rapidly developing branch of computer graphics known as *image based non-photorealistic rendering* (NPR). Perhaps the most well studied NPR problem is that of painterly rendering; the automated stylisation of imagery to give a hand-painted appearance. A number of these algorithms now exist capable of emulating a variety of styles, such as water-colour [1] and oil [2, 3, 4, 5, 6, 7].

Although the output of contemporary painterly rendering algorithms is often of high aesthetic quality, the usability of such algorithms is impeded by the plethora of low-level parameters that must be set in order to produce acceptable output. Many of these parameters are data dependent, for example determining the scale of image features to paint [3, 4]. The presence and fine-tuning of such parameters is necessary to retain generality of the algorithm, but can be difficult for the non-expert user to achieve. Furthermore, some algorithms [3, 6] seek to emulate a broad range of artistic styles using additional user configurable parameters. For example, [3] varies brush size, colour jitter, and stroke length to interpolate between pseudo "expressionist" and "pointillist" styles. Often these parameters can be time consuming to set — both due to their number, and due to their low-level nature, which can make them non-intuitive for inexperienced users to manipulate when aiming for a conceptually higher level effect (e.g. a gloomy painting, or an energetic, cheerful composition). Moreover, parameters can interact in complex ways leading to emergent behaviour within the painting that the user may not expect or understand. The end result is often a slow, iterative trial and error process before the user is able to instantiate their desired results.

This paper presents a solution to the problem of NPR parameter selection by framing the task as a goal-centred evolutionary search to be solved using a Genetic Algorithm (GA). We do not wish to eschew interaction, for this is often where artistic creativity is expressed when applying NPR algorithms (such techniques are better regard as tools, rather than black-box processes for creating artwork). Instead we draw inspiration from Sims [8] who adopted user supervision in his evolutionary artistic processes. Our system iteratively evolving a population of painterly renderings towards the user's aesthetic ideal, and on each evolutionary cycle presenting a sample of the population to the user for fitness evaluation. The user's evaluation affects the natural selection phase of the GA, and so affects the composition of subsequent generations of paintings. Similar architectures have also been applied in a other computer graphics domains; for example recent work facilitating the interactive evolution of pixel-vertex shaders [9] and animated screen-savers [10]. To facilitate timely interaction in our system we make use of a fast, segmentation based algorithm for painterly rendering. This algorithm not only draws upon existing painterly rendering literature (encapsulating many of the styles available using existing techniques), but also draws upon colour psychology to allow tonal variations that influence the emotional context or "mood" of the painting. This is achieved by mapping HSV colour transformations on to Russell's 2D "pleasure-arousal" emotional state space [11], and harnessing emotional state as a high level NPR parameter for painterly rendering [12].

The remainder of the paper is organised as follows. We begin by briefly surveying existing painterly techniques in Section 1.1. We then give detailed descriptions of our painting algorithm (Section 2) and interactive GA search (Section 3). We conclude with a gallery of results and discussion in Section 4.

1.1 Related Work

The majority of image based painterly rendering algorithms adopt the stroke-based rendering paradigm, generating paintings by compositing ordered lists of virtual "brush strokes" on a 2D virtual canvas. The development of such algorithms arguably began to gain momentum with Haeberli's semi-automatic "impressionist" painting system [13]. In Haeberli's system, stroke attributes such as colour or orientation were sampled from a source photograph whilst stroke size, shape and compositing order was set interactively by the user. Litwinowicz [2] was the first to propose a fully automated 2D painting algorithm, again focusing upon the impressionist style. Paintings were synthesised by aligning small rectangular strokes to Sobel edge gradients in the image and stochastically perturbing the colour of those strokes. Hertzmann later proposed a "coarse to fine" approach to painting using curved β-spline strokes . Spline control points were extracted by hopping between pixels in directions tangential to Sobel edges. The process operated at several discrete spatial scales, concentrating stroke detail in high frequency areas of the image. The β-splines used in this technique were later extended to active contours, enabling a relaxation based approach to curved stroke painting [14]. Other early painterly rendering algorithms such as [15] and [4] also made use of local image processing operators; placing strokes according to variance measures within a local window.

Our approach contrasts with these early painterly rendering algorithms in that we operate using multi-scale segmentation only, substituting image gradient and variance measures for region shape properties to guide stroke placement. Among the first to propose the use of segmentation algorithms for painting were Gooch *et al.* [5], who placed strokes along medial axes of segmented regions to produce painterly artwork. The benefits of their approach included a significant reduction in the number of brush strokes whilst preserving fine detail in the rendering. Segmentation was also used by [16, 17] to produce painterly abstractions. Notably these systems used a human gaze tracker to correlate level of detail in the painting with perceptually salient detail in the source image. An automatic system for salience adaptive painting, driven by machine learning rather than run-time interaction, was recently presented by Collomosse and Hall [7]. This work also applied evolutionary search techniques to NPR, harnessing GAs as a relaxation mechanism for automatically controlling level of detail in paintings within a single artistic style. By contrast we here apply GAs for style selection using interactive aesthetic evaluation, and as such, our work is also closely aligned with algorithms encompassing a range of visual styles selectable via user parameterisation. Hertzmann [3] claims expressionism, pointillism, impressionism and "abstract" styles through the variation of low level parameters such as stroke length. Similarly low-level parameters are used to tune the visual style of paintings in [6]. We have encompassed a similar gamut of rendering styles within our painterly framework.

2 Painterly Rendering Algorithm

In this section we briefly describe our fast multi-resolution technique for stylising photographs at interactive speeds. The scope of this paper is such that we have focused on the parameterisation of the algorithm and interested readers are directed to [12] for a more detailed description. The algorithm accepts a 2D photograph as input, and outputs a 2D painterly rendering of that photograph — the visual style of which is a function of eight user-configurable scalar parameters, $p_{1..8}$ (Figure 1) which are output by the evolutionary search step (Section 3).

We begin by creating a colour band-pass pyramid segmentation of the source image by applying the EDISON [18] algorithm at several spatial scales. This segmentation is computed only once, during system initialisation, to facilitate real-time interaction during the search process. To produce a painting, the pyramid layers are rendered in coarse to fine order. Segmented regions within each layer are painted using a combination of "interior" and "boundary" strokes; as

Param	Description
p_1	Colour jitter
p_2	Maximum hop angle
p_3	Region turbulence
p_4	Colour (pleasure)
p_5	Colour (arousal)
p_6	Stroke jaggedness
p_7	Stroke undulation
p_8	Region dampening

Fig. 1. Summary of the eight rendering parameters $p_{1..8}$ used to control visual style in our painting algorithm

we explain in the next subsection. For each layer, the "interior" strokes of all regions are first rendered, followed by the "boundary" strokes of all regions.

2.1 Interior and Boundary Stroke Placement

Brush strokes are formed using Catmull-Rom piecewise cubic splines, the control points of which are computed from the binary mask of each region as follows.

Interior Strokes. The interior of a segmented region is first filled using a modified boundary-fill algorithm that paints strokes tangential to the region's principal axis, obtained by computing the eigenvectors of pixel coordinates inside the region. Lines parallel to the principal axis are traversed, and strokes are started and terminated as region boundaries are encountered. The spacing between these traversal lines is proportional to stroke thickness. As each stroke is placed, control points are distributed uniformly over the stroke's length, and jittered via small translations to disguise the regularity of the stroke placement process. Stroke colour is set to the mean colour under the stroke, computed from the source image. This colour is subject to random perturbation, the magnitude of which forms one parameter of the rendering process, written p_1. Stroke thickness is set on a per region basis, in proportion to region area. In the case of very large regions, thickness is capped and strokes are painted horizontally (after [19]) to preserve natural appearance.

Boundary Strokes. The boundary of the segmented region is vectorised to produce a closed polygon. Points on the polygon are visited in order, and added to an initially empty "working set". Upon each point's addition, we sum the distance between all points in the working set to a line drawn between the first and last points in that set. If the distance is above a threshold (or no further points remain in the chain code), we output the most recently added point as a stroke control point. The working set is then emptied. A brush stroke is terminated, and a new stroke started, when the angle between adjacent control points rises above a preset threshold. In practice this threshold governs the typical length of brush strokes, and we allow this to vary between $0 - 50°$ as another of our rendering parameters (written p_2). The stroke must also be terminated if the colour of a new control point differs significantly from the mean colour of those already present in the stroke. Stroke thickness and colour are set as with the interior stroke placement process.

2.2 Rendering Parameters

In addition to the two rendering parameters (p_1, p_2) governing stroke placement (Section 2.1), we also incorporate the following six parameters that allow modulation of the painting's visual style. All parameters are summarised in Figure 1.

Region Turbulence. Flat expanses within paintings, for example sky or water, may be depicted in a variety of artistic styles. Our system encompasses a gamut of rendering styles ranging from the calm, serene washes of a watercolour to the energetic swirls of a Van Gogh oil or the chaotic strokes of a Turner seascape. We introduce similar effects by repeatedly performing boundary stroke placement (Section 2.1) subjecting region masks to morphological erosion prior to each iteration. The number of iterations is proportional to rendering parameter p_3. This has the effect of allowing boundary strokes to grow into the interiors of regions in an unstructured manner, so breaking up flat expanses in the painting.

Tonal Variation. Certain combinations of colours can evoke particular emotions, so helping to convey a particular mood to a composition. We have identified a number of cues from colour psychology, and mapped these to regions of Russell's 2D pleasure-arousal emotional space [11] — see Figure 2 (left). By specifying an emotional state (a point (p_4, p_5) in this space defined by two further rendering parameters), we allow the user to interactively vary the emotional ambiance or "mood" of the composition. Wright and Rainwater [20] have found the notion of happiness (pleasantness) to be primarily dependent on colour brightness (luminance), and to a lesser degree on saturation. Intuitively arousal corresponds to colour saturation, but can also be linked to hue. Wright and Rainwater's study has shown calmness to be blue-correlated [20], but according to Mahnke blue may also suggest depression and cold [21].

We have defined a number of transfer functions that operate upon hue, saturation and luminance as a mechanism for instantiating the colour heuristics we have distilled from the literature [20, 21]. The complex psychological theories underpinning colour and emotion generate non-linear mappings of hue, saturation and luminance variation to the pleasure-arousal space. We approximate these piecewise with a collection of linear transfer functions — different functions are applied in each of six regions of the space. Figure 2 (right) illustrates the boundaries of these regions, and the transfer functions used over the pleasure-arousal space. Functions $G(x)$ and $U(x)$ correspond to greying and un-greying (scaling saturation in proportion to x), while $D(x)$ and $L(x)$ correspond to lightening and darkening (scaling luminance in proportion to x). The operation of the latter function is capped for "boundary" brush strokes to prevent bleaching of fine detail. Care is taken in blending the constants of proportionality to prevent visible discontinuities near the boundaries defined over the pleasure-arousal space.

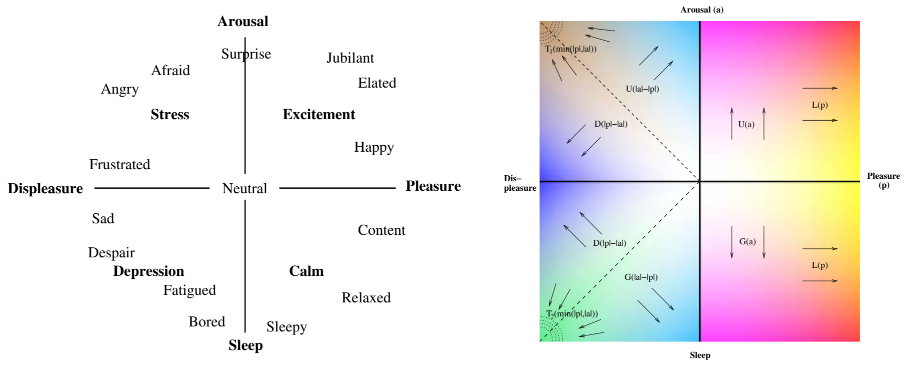


Fig. 2. Left: Russell's 2D pleasure-arousal space used to parameterise tonal variation in the painting. Right: False colour schematic illustrating the various colour transformations performed within regions of the pleasure-arousal space. Functions G, U, L, D, T_1 and T_2 are defined in Section 2.2.

Functions $T_1(x)$ and $T_2(x)$, indicated in Figure 2, are two special cases that encode hue variation consistent with aroused displeasure (anger) and apathetic displeasure (depression). Hue is manipulated via an RGB space transformation prior to saturation and luminance manipulations. In the former case $T_1(x)$, predominantly red colours are reddened and green (associated with calm) is reduced (in proportion to x). These effects combine with the saturation and luminance transformations already present to produce the combination of aroused reds and dismal darks that appear in psychological literature in association with anger. In the latter case $T_2(x)$ we increase the blue in proportion to x to generate a monotonous shift into the blue spectrum, associated with sadness and calm. Colours are also desaturated and darkened in accordance with transformations already present in that quadrant of the space.

Brush Stroke Style. We have introduced two parameters to control variation of stroke style in our system. These afford the user some control over stroke accuracy (p_6) and angularity (p_7) in the painterly rendering.

When rendering a brush stroke we create an arc-length parameterisation over the piecewise Catmull-Rom spline that smoothly interpolates its control points. To enhance the angularity or "jaggedness" of strokes we create an additional linear interpolation over the control points using the same arc-length parameterisation. Lineally interpolating between these two functions, in proportion to p_7, yields our desired style of stroke. We then introduce inaccuracies into the stroke placement process by inducing undulations in the trajectory of the stroke. Whilst rendering, we translate points on the stroke along their normal vectors — the distance a particular point is moved is set by a periodic function with frequency and amplitude proportional to p_6. Finally, we introduce a further parameter (p_8) to dampen the effects of undulation (p_6) on interior strokes. This can lead to visually chaotic stroke placements in the backgrounds of paintings that may or may not be desirable depending on the user's intended visual style.

3 Evolutionary Search for Parameter Selection

Given this parameterised rendering framework, the painting process is reduced to a search for the point in our parameter space $[p_1 p_2 ... p_8] \in \Re^8$) that corresponds to a painting expressing the target aesthetics of the user. Our system adopts a genetic algorithm (GA) search strategy. GAs are often cited as appropriate for exploring high dimensional parameter spaces as large regions of problem space can be covered quickly, and local minima (e.g. arising due to interactions.between painting parameters) are more likely to be avoided [22, 23]. We now describe the initialisation and iterative stages of our GA search.

3.1 Initialisation

We begin by initialising a fixed size population of individuals. We have opted for a population size of 1000 individuals, determined empirically to be a suitable trade-off between diversity the real-time processing constraints of our system. Each individual contains eight normalised scalar values that comprise the genotype of a particular painting. These values are seeded randomly in the initial generation.

3.2 Iterative Search

Genetic algorithms (GAs) simulate the process of natural selection by breeding successive generations of individuals through cross-over, fitness-proportionate re-production and mutation. In our implementation we terminate this iterative process when successive improvements in fitness become negligible (the change in both average and maximum population fitness over a sliding time window fall below a threshold). We now describe a single iteration of this process.

Interactive Evaluation. The first step in each iterative cycle is population evaluation. We wish to measure the proximity of each individual's phenotype to the user's "ideal" aesthetic. Specifically we require a mapping $M([p_1 p_2 ... p_8]) \mapsto f \in \Re$ where f is a normalised fitness score; higher values correspond to aesthetically superior paintings. As our aim is to assist the user in style specification it is not possible to write an automatic function for $M(.)$. Our objective is therefore twofold. First, to estimate the mapping function $M(.)$ through user interaction. Second, to search for the point $\underline{p} \in \Re^8$ such that:

$$\underline{p} = argmax_{\underline{i}} [M(\underline{i})] \tag{1}$$

Our approach is to sparsely evaluate $M(.)$ over a subset of the population, and use this data to extrapolate the behaviour of $M(.)$ over the entire population. We have designed a simple user interface, allowing us to prompt for the fitness of a given individual drawn from the population (so obtaining a sparse domain

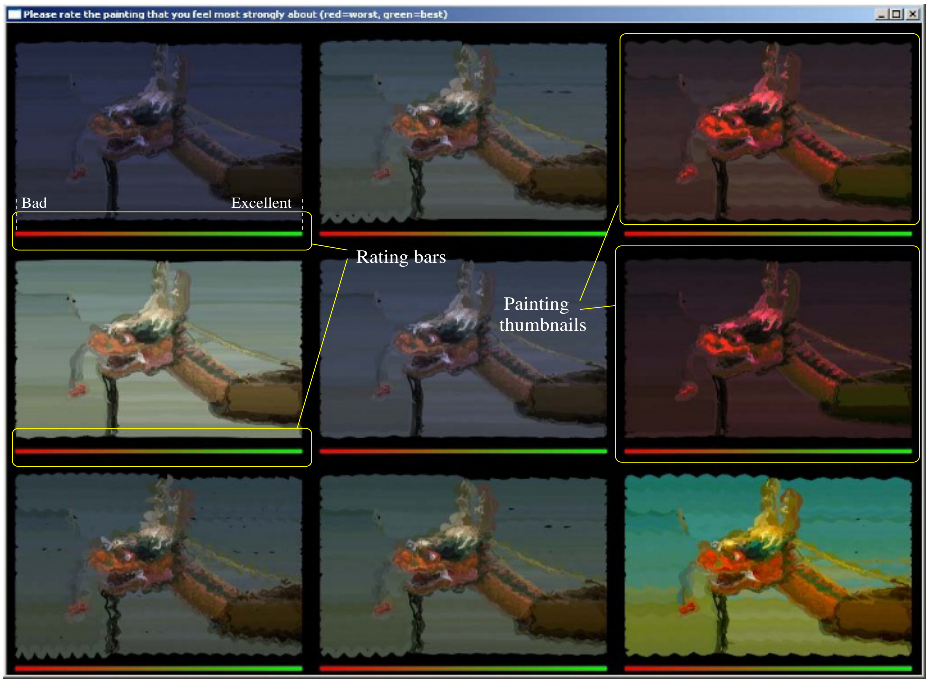

Please rate the painting that you feel most strongly about (red=worst, green=best)
Bad
Excellent
Rating bars
Painting thumbnails

Fig. 3. Snapshot of the interactive evaluation screen. The user is presented with thumbnails of the highest ranking 9 paintings and asked to rate one by clicking with the mouse. Depending on the horizontal location of the click within the thumbnail, a fitness score [-1,1] is assigned to the chosen rendering. This snapshot shows images from the first generation of paintings — hence the diverse selections available.

sample of $M(.)$). The user is supplied with a graduated colour bar, and asked to rate the aesthetics of the painting rendered from a given individual on a continuous scale spanning red (bad), amber (neutral) and green (excellent) — see Figure 3. Manually evaluating the entire population on each iteration would be impractical, and to reduce user load we request evaluation of only one individual per generation. The user is presented with the nine fittest individuals from the previous iteration, and asked to rate the individual that they feel most strongly about. Note that in the first iteration individuals are deemed to exhibit equal fitness (see equation 2) and so are chosen from the population at random.

We use a Gaussian "splatting" technique to encode the results of our sparse user interactions, and transform these into a continuous estimate for $M(.)$. Each time a user evaluates an individual we obtain a point $\underline{q}$ and a user fitness rating $U(\underline{q}) = [-1, 1]$. These data are encoded by adding a Gaussian to a cumulative model, built up over successive user evaluations. Each Gaussian distribution is centred at point $\underline{q}$, and multiplied by the factor $U(\underline{q})$. We assume the integral under the Gaussian to be well approximated by unity in space $\Re^8 \in [0, 1]$, and so infer the continuous function $M(.)$ as:

$$M(\underline{p}) = 0.5 + \begin{cases} 0 & \text{if } N = 0, \\ \frac{1}{2N} \sum_{i=1}^{N} U(\underline{q}_i) G(\underline{p}, \underline{q}_i, \sigma) & \text{otherwise} \end{cases} \tag{2}$$

where $\underline{p}$ is an individual to be evaluated in the current generation, $\underline{q}_i$ are individuals evaluated by the user in the previous N iterative cycles, and $U(\underline{x})$ is the user's score of a given genotype $\underline{x}$. The function $G(\underline{x}, \underline{\mu}, \sigma)$ denotes a Gaussian distribution with mean $\underline{\mu}$ and standard deviation σ, evaluated at $\underline{x}$. The standard deviation σ governs the locality in problem space over which a single user evaluation holds influence. We have used the value $\sigma = 0.1$ for all of the results presented here. Equation 2 provides us with an estimate for $M(.)$ defined over the entire problem space, which we then apply to evaluate the whole population.

Selection and Propagation. Once the current population has been evaluated, pairs of individuals are selected and bred to produce the next generation of painting solutions. Parent individuals are selected with replacement, using a stochastic process biased toward fitter individuals. A single offspring is produced from two parents by way of stochastic cross-over and mutation operators. Each of the eight parameters that comprise the genome of the offspring has an equal chance of being drawn from either parent. Mutation is implemented by adding a random normal variate to each of the eight parameters. These variates have standard deviations of 0.1, i.e. 97% of mutations will produce less than ± 0.3 variation in a particular rendering parameter.

4 Results and Conclusion

We have tested our system on a wide range of images, a give representative examples in Figure 5. In Figures 5a we show single photograph (*BIGBEN*) evolved into "abstract" and "expressionist" styles reminiscent of those presented in [3, 24]. Convergence took 15 and 17 mouse clicks respectively — less than one minute

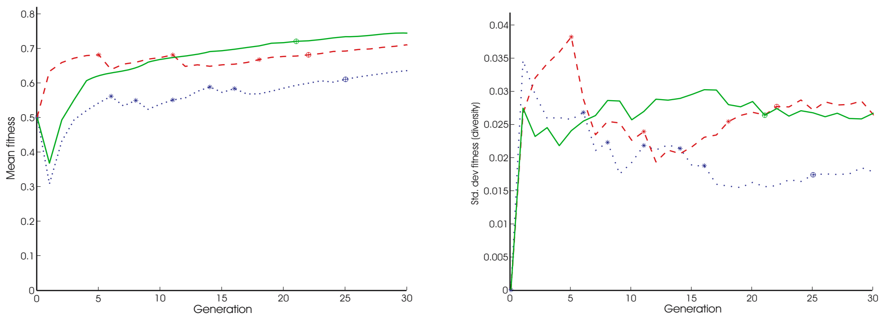


Fig. 4. Population statistics corresponding to the evolution of the paintings shown in Figure 5c_1 (blue, dotted), Figure 5c_2 (red, dashed), Figure 5c_3 (green, solid) respectively. The + symbol indicates algorithm termination. $*$ indicates a negative fitness rating from the user.

of user time. Furthermore, our segmentation based painting algorithm did not require technical parameters (e.g. scale or low-pass kernel size [3]) to be specified explicitly by the user. Figures 5e$_1$, e$_2$ give examples of further painterly styles, contrasting two different stroke placement styles encompassed by our system. The first is reminiscent of the "impressionist" style paintings generated by [2], the latter the impasto style oil paintings generated by [7]. Figure 5b gives a further examples of the broad classes of image handled by our system. Figure 5c demonstrates a single photograph (*DRAGON*) evolved into three distinct visual styles. A non-expert was asked to use our system to create paintings depicting high-level concepts such as anger (Figure 5c$_1$), cheerfulness (Figure 5c$_2$) and despair (Figure 5c$_3$). Graphs recording the mean population fitness, and standard deviation (diversity) during the evolution of these paintings are also supplied in Figure 4. Convergence took between 20 to 25 generations before the termination criteria was triggered. In Figure 4 we have forced evolution to continue beyond 30 generations, however improvements in mean fitness beyond the automated termination point are negligible.

To evaluate the usability of our system we developed an alternative low-level interface using sliders to independently control $p_{1..8}$. Users were asked to produce identical renderings to those previously generated using our GA. Users were usually able to reproduce results, but required five or six minutes of experimentation (and several hundred mouse clicks) before doing so — approximately five times longer than when using our GA goal-based search.

When working with our system, we have found that users will often focus on a particular aesthetic property of the painting and focus on the improvement of that property. For example, users might address the issue of edge detail over, say, the style of the in-filled background. Often these properties have no direct mapping onto individual rendering parameters, providing some explanation of the timing improvements of a top-down goal seeking approach to style selection over a bottom up configuration of low-level painting parameters. The impact of this behaviour can be observed in the graph of Figure 4 (left). Gradual increases in painting "fitness" are observed over time, interrupted by short-lived dips. These dips become less pronounced as the generation count increases. We have found the presence of dips to correlate with the issuing of negative ratings by users; typically these are issued when a user has refined one aspect of the painting to their liking, and begun to address a further aspect of the composition that they had so far neglected. By neglecting refinement of the latter aspect, a false representation of the user's "fitness function" $M(.)$ (see Section 3.2) has been conveyed to the system and encoded in the Gaussian distribution model. This requires user correction, often in the form of rating penalisation.

Throughout our work we have assumed the user has an ideal painting in mind, and wishes to express instantiate that ideal through NPR. An alternative application of our system might be in style exploration, where the user has no well-developed goal state in mind. Early indications are that the guided search provided by our system may be suitable for such activities. However if the user

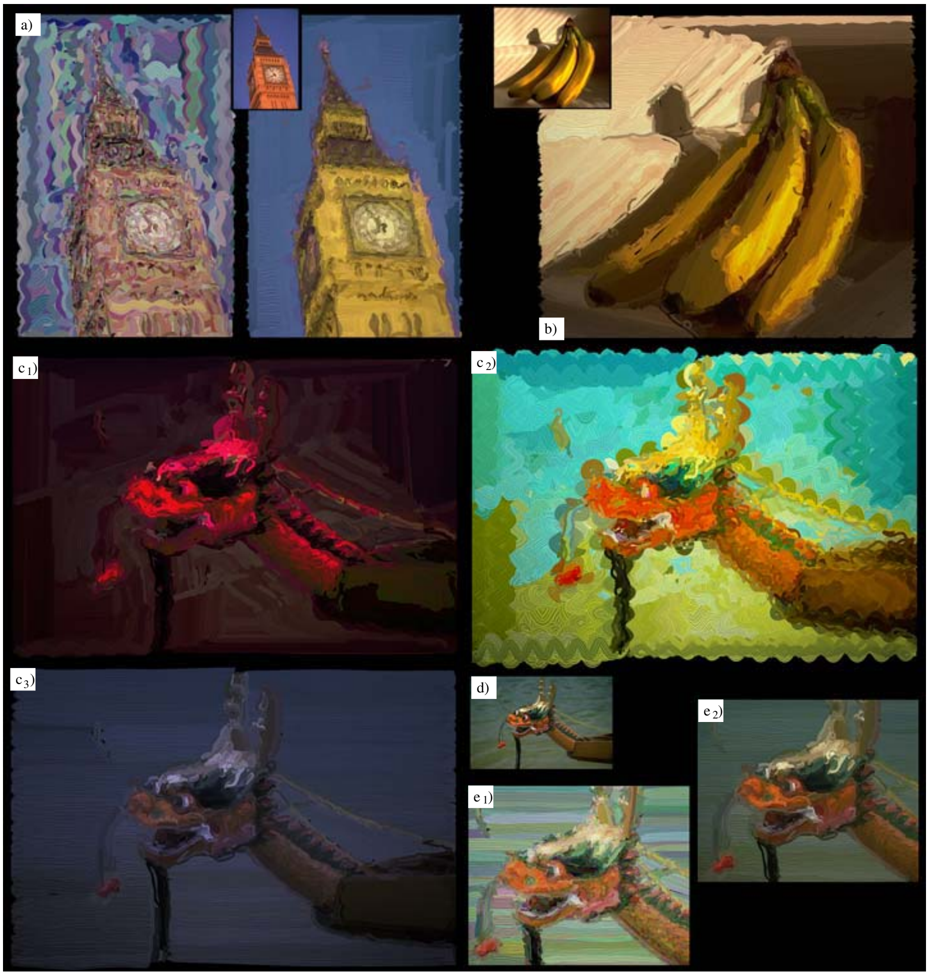


Fig. 5. A gallery of painterly renderings produced by our system, original images inset

substantially revises their aesthetic ideals late in the search, the reduced population diversity can require tens of iterations before the user is able to explore new regions of the problem space. If our system were to be used in this manner, we would suggest increasing the standard deviation of the variates used during mutation to maintain population diversity further into the search.

Acknowledgements

We are grateful for the assistance of users who participated in evaluation of the GA interface, and also for the contribution of Maria Shugrina in development of the colour model.

References

1. Curtis, C., Anderson, S., Seims, J., Fleischer, K., Salesin, D.H.: Computer-generated watercolor. In: Proc. ACM SIGGRAPH. (1997) 421–430
2. Litwinowicz, P.: Processing images and video for an impressionist effect. In: Proc. ACM SIGGRAPH, Los Angeles, USA (1997) 407–414
3. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: Proc. ACM SIGGRAPH. (1998) 453–460
4. Shiraishi, M., Yamaguchi, Y.: An algorithm for automatic painterly rendering based on local image approximation. In: Proc. ACM NPAR Sympos. (2000) 53–58
5. Gooch, B., Coombe, G., Shirley, P.: Artistic vision: Painterly rendering using computer vision techniques. In: Proc. ACM NPAR Sympos. (2002) 83–90
6. Hays, J., Essa, I.: Image and video based painterly animation. In: Proc. ACM NPAR Sympos. (2004) 113–120
7. Collomosse, J.P., Hall, P.M.: Genetic paint: A search for salient paintings. In: proc. EvoMUSART (at EuroGP), Springer LNCS. Volume 3449. (2005) 437–447
8. Sims, K.: Artificial evolution for computer graphics. In: Proc. ACM SIGGRAPH. Volume 25. (1991) 319–328
9. Ebner, M., Reinhardt, M., Albert, J.: Evolution of vertex and pixel shaders. In: LNCS (in Proc. EuroGP'05). Volume 3447., Springer-Verlag (2005) 261–270
10. Draves, S.: The electric sheep screen-saver: A case study in aesthetic evolution. In: LNCS (in Proc. EvoMUSART'05). Volume 3449., Springer-Verlag (2005) 458–467
11. Russell, J.A.: Reading emotion from and into faces: Resurrecting a dimensional-contextual perspective. In Russel, J.A., Fernández-Dols, J.M., eds.: The Psychology of Facial Expression. Cambridge University Press (1997) 295–320
12. Shugrina, M., Betke, M., Collomosse, J.P.: Empathic painting: Interactive stylization using observed emotional state. In: Proc. ACM NPAR Sympos. (2006)
13. Haeberli, P.: Paint by numbers: abstract image representations. In: Proc. ACM SIGGRAPH. Volume 4. (1990) 207–214
14. Hertzmann, A.: Paint by relaxation. In: Proc. Computer Graphics Intl. (CGI). (2001) 47–54
15. Treavett, S., Chen, M.: Statistical techniques for the automated synthesis of non-photorealistic images. In: Proc. 15th Eurographics UK Conference. (1997) 201–210
16. DeCarlo, D., Santella, A.: Abstracted painterly renderings using eye-tracking data. In: Proc. ACM SIGGRAPH. (2002) 769–776
17. Santella, A., DeCarlo, D.: Visual interest and NPR: an evaluation and manifesto. In: Proc. ACM NPAR Sympos. (2004) 71–78
18. Christoudias, C., Georgescu, B., Meer, P.: Synergism in low level vision. In: 16th Intl. Conf. on Pattern Recognition. Volume 4. (2002) 150–155
19. Kolliopoulos, A.: Image segmentation for stylized non-photorealistic rendering and animation. Master's thesis, Univ. Toronto (2005)
20. Wright, B., Rainwater, L.: The meaning of colour. Journal of General Psychology **67** (1962)
21. Mahnke, F.: Color, Environment, and Human Response. Van Nostrand Reinhold (1996)
22. de Jong, K.: Learning with genetic algorithms. Machine Learning **3** (1988) 121–138
23. Holland, J.: Adaptation in Natural and Artificial Systems. 1st edn. Univ. Michigan Press (1975) ISBN: 0-472-08460-7.
24. Hertzmann, A., Perlin, K.: Painterly rendering for video and interaction. In: Proc. ACM NPAR Sympos. (2000) 7–12

Robot Paintings Evolved Using Simulated Robots

Gary Greenfield

Mathematics & Computer Science,
University of Richmond,
Richmond VA 23173, USA
ggreenfi@richmond.edu
http://www.mathcs.richmond.edu/∼ggreenfi/

Abstract. We describe our efforts to evolve robot paintings using simulated robots. Our evolutionary framework considers only the initial positions and initial directions of the simulated robots. Our fitness functions depend on the global properties of the resulting robot paintings and on the behavior of the simulated robots that occurs while making the paintings. Our evolutionary framework therefore implements an optimization algorithm that can be used to try and help identify robot paintings with desirable aesthetic properties. The goal of this work is to better understand how art making by a collection of autonomous cooperating robots might occur in such a way that the robots themselves are able to participate in the evaluation of their creative efforts.

1 Introduction

Open Problem #3 of McCormack's five open problems in evolutionary music and art (EMA) [1] requires one, "To create EMA systems that produce art recognized by humans for its *artistic* contribution (as opposed to any purely technical fetish or fascination)." The recent publicity garnered by the *robot paintings* of Moura, Ramos, and Pereira that resulted from their ARTSBOT (ARTistic Swarm roBOTS) Project might at first glance be seen as a solution to McCormack's third open problem since the paintings are described on the web (see http://alfa.ist.utl.pt/ cvrm/staff/vramos/Artsbot.html) as "artificial art," and in print as "non-human art" [2] or "symbiotic art" [3]. Note that here the symbiosis is intended to be between human and robot. The site http://www.lxxl.pt/artsbot/ where the images of the robot paintings with the best resolution can be found also provides a "Symbiotic Art Manifesto" written by Moura and Pereira.

It is unfortunate that some of the hyperbole associated with the ARTSBOT project detracts from what is potentially a promising new development in evolutionary art. At the center of the ARTSBOT Project lies an implementation of a collective robotics art making *system* to create what are known as *swarm paintings*. The ARTSBOT team reveals this by saying [4] — to paraphrase and polish slightly — that the artworks are made by "a swarm of autonomous robots, that ‘live’ [by] avoiding simply [executing] streams [of commands] coming from an external computer, [and] instead actually co-evolve within the canvas [environment]; acting [by] laying ink according to simple inner threshold stimulus response functions, [while] simultaneously reacting to the chromatic stimulus present in the canvas environment [left by other robots], as well as by the distributed feedback, [that] affect[s] their future collective behavior.” We note that ARTSBOT was one of the few *collective* robotics entries in the most recent international ArtBot art exhibition for “robotic art and art-making robots” (see http://artbots.org/2004/participants/). Moura and Pereira claim that they have created organisms that generate drawings without any intervention on their part thereby creating “a new kind of art” based on a paradigm of non-human autonomous entities that allow personal expression by human artists to be abandoned [3]. Perhaps this somewhat of an exaggeration. Since the controllers for their robots were not evolved, ARTSBOT is not an *evolutionary* art system, but rather an art making system consisting of human programmed autonomous agents that reside and function in an artificial ecosystem. There is a close connection here between *stigmergy* [5] — the situation where autonomous agents alter their environment either accidentally or on purpose in such a way that they influence other agents to perform actions that achieve an objective such as nest building — and swarm painting. The principal difference is that stigmergy is usually associated with a clearly defined task or objective while swarm painting is usually associated with the more poorly defined objective of producing aesthetic imagery.

While in our opinion the question of whether or not the ARTSBOT robot paintings are more than what McCormack referred to as a “technical fascination” has not yet been satisfactorily answered, what is most significant to us is the fact that ARTSBOT does not address McCormack’s penultimate challenge, Open Problem #5, which requires one: “To create artificial ecosystems where agents create and recognize their own creativity.” In this paper using *simulated* collective robotics and taking for motivation the penultimate problem of how autonomous robots engaged in making swarm paintings might eventually go about learning to *recognize* their own creativity, as a first step we investigate an evolutionary framework that is designed to show how simulated robots might be able to formulate ways to *evaluate* the aesthetic quality of their paintings. Unlike the aesthetic evaluation system for agent produced art studied by Saunders and Gero where each agent produced its own paintings and the evaluation model was based on social dynamics [6], we consider an aesthetic evaluation system where the collective agents are given shared access to a set of image evaluation parameters which can then be used either individually or collectively to modify the image making process. To help understand the consequences of our design, we consider what effect different types of computations made using our set of evaluation parameters have on our robot paintings.

This paper is organized as follows. In section two we provide some background on the use of swarms and the non-interactive genetic algorithm for image making. In section three we give the specifications for our simulated robots. In section four we describe how their controllers work. In section five we present our evolutionary framework. In section six we define our set of image evaluation parameters and then proceed to give examples of some of the robot paintings we evolved using various fitness functions formulated based on these parameters. In section seven we consider the implications of our work for the problem of how robot swarms might go about evaluating their creative efforts. In section eight we offer our summary and conclusions.

2 Background

The notion of swarm paintings was first introduced in an image processing paper by Ramos that was contributed to an ant colony optimization (ACO) conference [7]. Related work appeared in [8] and [9]. In this problem domain the use of the interactive user-guided evolution paradigm that was originally proposed by Sims [10] (i.e. the interactive genetic algorithm) was first studied by Aupetit et al [11]. They investigated an ant colony simulation where the virtual ants deposited and followed color – the *scent* — while exploring a toroidal grid in order to produce "ant paintings." Greenfield [12] non-interactively evolved ant paintings by evolving the genomes required for governing the behaviors of the virtual ants using fitness functions. His observation that only elementary techniques were needed to measure ant exploration and ant cooperation capabilities offers hope that relatively simple behavioral assessment parameters can be used to help identify increased image complexity or well organized image compositions in other evolutionary swarm painting scenarios. The use of the (non-interactive) genetic algorithm in evolutionary art was first considered by Baluja et al [13]. Using this technique for evolving two-dimensional imagery, interesting results have been obtained by Greenfield [14] using co-evolution and the image generation method known as "evolving expressions", by Machado and Cardosa [15] using neural nets, and by Bentley [16] in order to identify cellular automata "patterns." In general, the question of how to evaluate aesthetics on the basis of scientific principles and computational methodologies is a difficult one. To sample several different author's thoughts on the matter and help gauge the scope of the debate see [17, 18, 19, 20, 21].

3 S-Robot Specification

The design of our simulated robots, or S-robots, is loosely based on a software model for Khepera robot simulation by Harlan et al [22]. An S-robot is a virtual circular entity with four binary valued *proximity* sensors together with a three-channel color sensor. Three of the proximity sensors are located at the front of the S-robot and the fourth is located at the rear. The forward and backward sensors scan a field 120° wide and the two side sensors scan a field 45° wide in such a way that there is a 15° overlap with the forward sensor. Thus the forward facing 'field of vision" is from −90° to +90° up to a distance of twenty units and the rear facing field of vision is from −60° to 60° also up to a distance of twenty units. Proximity sensors detect other robots and environmental obstacles or boundaries but do not distinguish between the two. The color sensor is mounted directly beneath center (r_x, r_y) of the S-robot. The S-robot's forward direction is determined by the *unit* vector (d_x, d_y). For all of the images shown here, the robot's two pens were operated synchronously so that either both were up or both were down. The reason for this was so that when the S-robot was mark making, the pen colors could be chosen so that that the mark had an automatic built-in highlight. An S-robot's painting mark is five units wide. An S-robot can swivel (i.e rotate in place) $10°$ clockwise or counterclockwise per clock cycle and can move v units per clock cycle, $-1 \leq v \leq 1$, in either the forward or backward direction in accordance with the sign of v. The S-robot roams on an $n \times m$ unit gridded world.

4 S-Robot Controllers

The "onboard computer" for an S-robot is an interrupt driven controller whose job is to place a sequence of commands in an execution queue, sleep until the queue is empty, and then plan and load the S-robot's next sequence of commands when it is awoken. An S-robot is autonomous because it can place commands in the queue to request sensors readings so that when it is awoken it can perform actions based on these sensor values. The controller loads commands of the form <mnemonic> <argument> where the mnemonic is chosen from the list:

MOV	Move
SWI	Swivel
SPD	Set Speed
SNP	Sense Proximity Vector
SNC	Sense Color Vector
PUP	Pen Up
PDN	Pen Down

Only the MOV, SWI, and SPD commands actually make use of the argument, in all other cases it is treated as a dummy argument. By having the controller indicate how far it wants the S-robot to travel, or how many degrees it wants the S-robot to swivel, the burden of timing shifts to the simulator itself. The simulator calculates how many clock cycles these actions will take so that it can manage the discrete event scheduling, synchronize the movements of all the S-robots, detect collisions, and update the sensors accordingly.

While in the future we would like to evolve the controllers themselves, in this paper we make use of two controllers that we wrote ourselves in order to consider how the cooperation between two S-robots was affected by their initial placement and direction headings. Each of our controllers has four pre-planned painting sequences it can load into the queue. For ease of managing simulated evolution and evaluating the results, at run time we made only one of the four painting sequences available to each controller. The four sequences can produce an elongated double hooked curve, a wedge, a segment of a spiral, and a zigzag

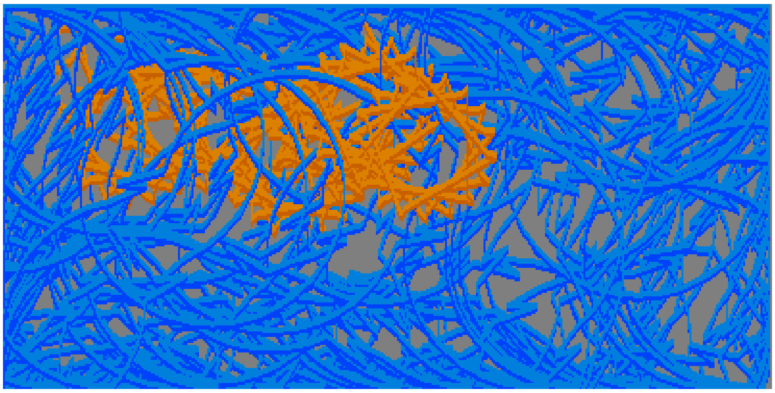


Fig. 1. Two S-robots using different controllers and different painting motifs. Note that one of the S-robots did most of its painting by leaving its pens *down* while executing a back-up and swivel sequence following boundary collisions.

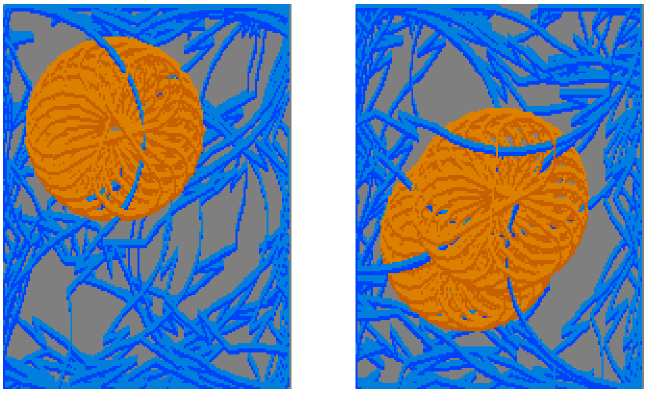


Fig. 2. Images of two S-robots painting with, and without, exhibiting robot interaction. On the left, the S-robot painting the closed figure is oblivious to its companion, while on the right it collides with its companion and gets bumped into a new painting trajectory.

motif. Figure 1 shows an early S-robot test painting made using two S-robots where one used the double hooked curve to draw closed figures and the other used the zigzag sequence as it tried to roam more freely over the canvas. The latter left the pens down during a back-up obstacle avoidance sequence which explains the appearance of the long curving trails.

We now describe our two controllers. Controller A always first checks the forward sensor. If it is clear, it queues the assigned painting command sequence followed by commands to swivel, move a short distance away, and take a proximity reading. If the forward sensor is set, but the backward sensor is clear, it queues a back-up sequence followed by swivel sequence and again requests a proximity reading. Otherwise, having concluded it is boxed in, it swivels and tries to move only a short distance away before taking a new proximity reading. Controller B, on the other hand, can be set up so that it uses the color channel sensors to search either for areas of the canvas that have not yet been painted or for areas that have been painted by one of its companions. Whenever it locates pixels of the type it is searching for, it queues the assigned painting command sequence followed by a swivel sequence, otherwise it swivels and moves a short distance from its present location. In both cases it again queues a color reading. Figure 2 shows what happens when an S-robot with an A controller that is drawing a closed figure gets bumped off course when an S-robot with a B controller that is trying to fill in unpainted canvas gets too close.

5 Evolutionary Framework

The S-robot paintings described below were all painted on 200×200 canvases. The S-robots were permitted to paint for 150,000 clock cycles. The genome for an individual S-robot is the vector (s_x, s_y, d) where (s_x, s_y) is its initial position and d is its initial true compass heading, $-180 \leq d < 180$. For a collection, or swarm, of N S-robots the genome g is the concatenation of the genomes of the individual S-robots. Thus g is a vector with $3N$ components. The point mutation operator applied to g displaces each component of g by a small amount, while the crossover operator applied to genomes from two swarms implements the usual *uniform* crossover operator for two vectors with the same number of components.

Our evolutionary framework uses a population of size $P = 16$. Some evolutionary runs set the number of S-robots at $N = 2$ while others use $N = 4$. For each of $G = 30$ generations, the painting made by the swarm of S-robots with genome g is assigned fitness F_g using one of the calculation methods described below. Then the $P/2$ least fit genomes are discarded and $P/4$ breeding pairs are formed by cloning from the pool of $P/2$ survivors. Breeding within each pair is performed using crossover. Finally all P genomes are subjected to point mutation. Thus an evolutionary run considers $G \cdot P = 30 \cdot 16 = 480$ S-robot paintings. The painting associated with the most fit genome is logged after every five generations. Since point mutation is applied to every genome in the population at the conclusion of every generation, the implicit genetic algorithm is non-elitest and therefore the generation in which the most fit genome will appear during the course of a run cannot be predicted in advance.

6 The S-Robot Fitness Calculation

When a group of N S-robots is making an S-robot painting, the following data is collected: n_p, the number of squares of the grid that were painted; n_b, the number of times an S-robot reacted to the situation where the forward proximity bit was set but the backward proximity bit was clear; n_s, the number of times an S-robot reacted to the situation where the forward and backward bits were both set; and n_c, the number of times an S-robot was successful at color sensing. Figure 3 shows an example of the improvement in image “complexity” that occurred over

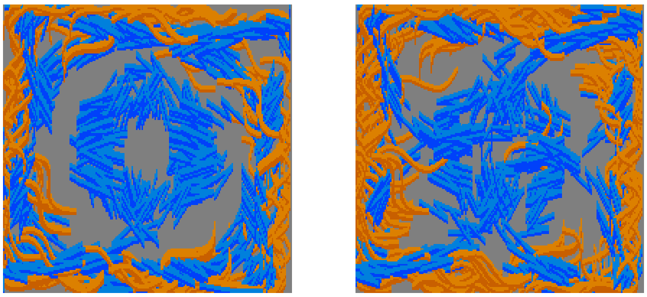


Fig. 3. S-robot paintings where fitness was determined by the two S-robot's ability to cover the canvas. The image on the left is the most fit image in the initial randomly generated population, the image on the right is the most fit image after ten generations.

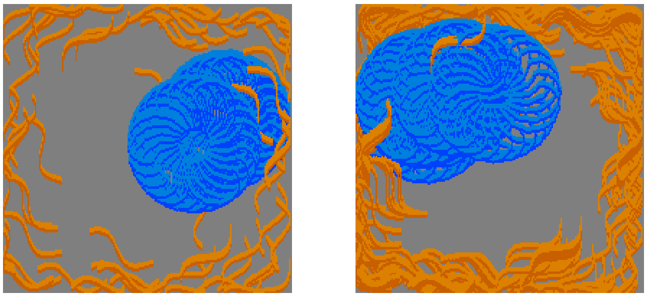


Fig. 4. Two S-robot paintings where image fitness was determined using a linear combination of the S-robot behavioral assessment terms. The goal is to evolve a composition. The image on the left is the most fit image from the original population, the image on the right is the most fit image from the twentieth generation.

time simply by letting $F_g = n_p$, thereby ensuring that the proportion of canvas that was painted was optimized. Figure 3 shows a comparison of the most fit image from the initial randomly generated genome population with the most fit image after ten generations.

Figure 4 shows two S-robot paintings obtained using the fitness function given by $F_g = n_p - n_b + 100n_s + 1000n_c$. Over time evolution causes the canvas to fill in more and locates the closed figure in such a way that maximal S-robot interaction can occur. Figure 5 shows the two most fit S-robot paintings after fifteen and thirty generations from a run using fitness given by $F_g = n_p - 100n_s + 1000n_c$. They show two different "solutions" to the optimization problem posed. One exhibits mutual following behavior by the two S-robots and the other exhibits avoidance behavior since one S-robot retreats to a corner and lingers there. Figure 6 shows the synergy that resulted when the fitness function $F_g = n_s n_c$ was used and the color sensing robot was initialized to seek the paint trails of its companion. Finally, Figure 7 shows an example using fitness function $F_g = n_s n_c + n_p n_b$, which adds a new term to the previous fitness

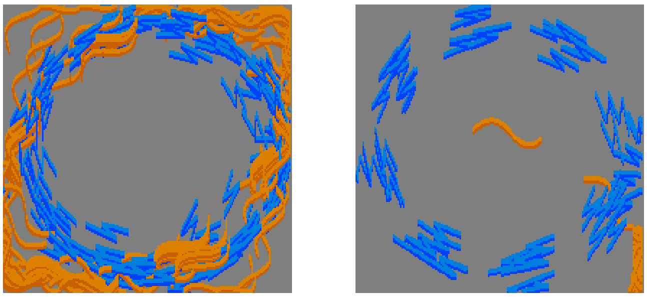


Fig. 5. S-robot paintings from the fifteenth and thirtieth generations obtained from a run that used a fitness function that maximized the assessment terms n_p and n_c, while minimizing n_s

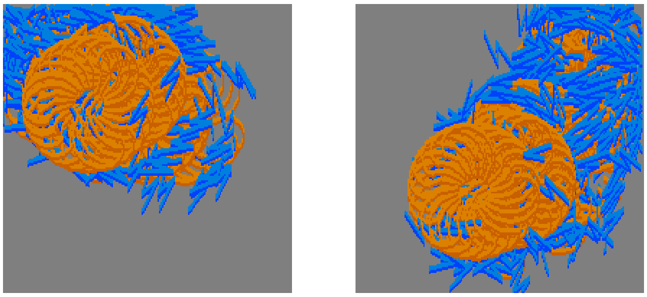


Fig. 6. S-robot paintings from the tenth and twentieth generations obtained from a run that used a fitness function that maximized S-robot interaction by using a product of the behavioral assessment terms n_b and n_c

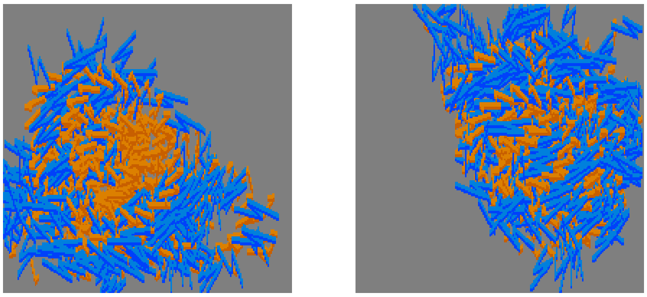


Fig. 7. S-robot paintings from the fifth and twentieth generations obtained from a run that used a fitness function with interaction terms to maximize both S-robot interaction and canvas coverage

function in order to increase canvas coverage in an effort to compensate for the fact that one of the S-robots is now being restricted to making a smaller mark when it paints. We believe these examples help support our contention that it is possible to impose a *style* on robot paintings by carefully devising the fitness functions. That is, following a relatively brief period of experimentation to discover how weighting and combining the parameters affects S-robot paintings, one can become reasonably competent at formulating fitness functions using the parameters in such a way that the evolved imagery will matching one's own aesthetic tastes.

7 On Autonomous Fitness Calculation

The previous section showed how the evolution of our S-robot paintings occurs by using optimization to select the initial configurations of the individual S-robot settings. This optimization treats the fitness calculation as a computation that assigns an aesthetic value to each painting by the swarm of S-robots that created it. Even though it would be a very time consuming process, we feel it is important to make the observation that this fitness calculation could be performed by the S-robots themselves, because we believe that any collection of robots that is engaged in evaluating or recognizing their own creativity would need to include some kind of aesthetic evaluation capability such as ours. Of course, to fully implement the protocol that our S-robots would need to follow in order to achieve this aesthetic evaluation goal, the functionality of the S-robots would need to be enhanced so that they could exchange data with one other, make use of a pseudo random number generator, and have their initial position and heading correctly calibrated. Assuming this were done, an outline of the protocol would be:

1. S-robots save their initial positions and headings.
2. While the painting is being executed, S-robots save information needed to *collectively* calculate image fitness.
3. Designated S-robot traverses entire canvas to determine global statistics needed for fitness calculation (e.g. paint coverage of canvas).
4. S-robots share data in such a way that each is able to calculate image fitness.
5. S-robots compare current fitness value to their saved fitness values to decide, if necessary, which two of *their* saved genomes to cross, before mutating their genomes for the next painting.
6. S-robots are placed on a new canvas with the correct desired initial positions and headings.

It should be clear that it would not be too difficult to design more sophisticated protocols for robot genomes involving controller settings, planning algorithms, or painting sequences in addition to the initial configuration data.

8 Summary and Conclusions

We considered the problem of how to evolve swarm paintings. We did so by developing an evolutionary framework using simulated autonomous mark-making robots. To use the non-interactive genetic algorithm within this framework, we introduced global image assessment parameters and local behavioral assessment parameters that could be used for formulating fitness functions to evaluate, or rank, images on the basis of criteria intended to identify aesthetically interesting paintings. Even though evolution was only able to control the initial placement and positioning of the robots, we gave examples to show how the use of different fitness functions could affect the aesthetic qualities of the robot paintings. We also explained how, in principle, our evolutionary fitness scheme could be managed by the robots themselves. We believe this represents a first step towards reaching the eventual goal of having autonomous robots collectively evaluate and recognize their own creative efforts.

References

1. McCormack, J. (2005): Open problems in evolutionary music and art. In Rothlauf, F. et al (eds.) Applications of Evolutionary Computing, EvoWorkshops 2005 Proceedings, Springer-Verlag Lecture Notes in Computer Science, LNCS 3449, 428–436.
2. Moura, L. and Ramos, V. (2002): Swarm paintings — nonhuman art. In Maubant, J. et al (eds.), Architopia: Book, Art, Architecture, and Science, Institut d'Art Contemporain, Lyon/Villeurbanne, France, 5–24.
3. Moura, L. and Pereira, H. (2004): Man + Robots: Symbiotic Art. Institut d'Art Contemporain, Lyon/Villeurbanne, France.
4. Ramos, V. (2003): Self-organizing the abstract: canvas as a swarm habitat for collective memory, perception and cooperative distributed creativity. In Rekalde, J. et al (eds.) First Art & Science Symposium, Models to Know Reality, Bilbao, Spain, 59.
5. Theraulaz, G. and Bonabeau, E. (1999): A brief history of stigmergy. In Artifical Life **5:2** 97–116.
6. Saunders, R. and Gero, J. (2001): Artificial creativity: a synthetic approach to the subject of creative behavior. In Gero, J. (ed.) Proceedings of the Fifth Conference on Computational and Cognitive Models of Creative Design, Key Centre of Design Computing and Cognition, Sydney, Australia, 113–139.
7. Ramos, V. and Almeida, F. (2000): Artificial ant colonies in digital image habitats - a mass behaviour effect study on pattern recognition. In Dorigo, M. et al (eds.) Proceedings of ANTS'2000 - 2nd International Workshop on Ant Algorithms (From Ant Colonies to Artificial Ants), Brussels, Belgium, 113–116.
8. Ramos, V. (2002): On the implicit and on the artificial - morphogenesis and emergent aesthetics in autonomous collective systems. In Maubant, J. et al (eds.), Architopia: Book, Art, Architecture, and Science, Institut d'Art Contemporain, Lyon/Villeurbanne, France, 25–27.
9. Ramos, V. and Merelo, J. (2002): Self-organized stigmergic document maps: environment as a mechanism for context learning. In Alba, E. et al. (eds.) AEB2002, First Spanish Conference on Evolutionary and Bio-Inspired Algorithms, Mérida, Spain, 284–293.
10. Sims, K. (1991): Artificial evolution for computer graphics. In Computer Graphics **25** 319–328.

11. Aupetit, S., Bordeau, V., Slimane, M., Venturini, G. and Monmarche, N. (2003): Interactive evolution of ant paintings. In McKay, B. et al (eds), 2003 Congress on Evolutionary Computation Proceedings, IEEE Press, 1376–1383.
12. Greenfield, G. (2005): Evolutionary methods for ant colony paintings. In Rothlauf, F. et al (eds.) Applications of Evolutionary Computing, EvoWorkshops 2005 Proceedings, Springer-Verlag Lecture Notes in Computer Science, LNCS 3449, 478–487.
13. Baluja, S., Pomerleau, D. and Jochem, T. (1994): Towards automated artificial evolution for computer-generated images. In Connection Science **6** 325–354.
14. Greenfield, G. (2002): On the co-evolution of evolving expressions. In International Journal of Computational Intelligence and Applications **2:1** 17–31.
15. Machado, P. and Cardoso, A. (1998): Computing aesthetics. In Oleiveira, F. (ed.) Proceedings XIV-th Brazilian Symposium on Artificial Intelligence SBIA'98, Porto Allegre, Brazil, Springer-Verlag, LNAI Series, New York, NY, 219–229.
16. Bentley, K. (2002): Exploring aesthetic pattern formation. In Soddu, C. (ed.), Proceedings of the Fifth International Conference of Generative Art (GA 2002), Milan, Italy. (See http://www.generativeart.com/papersGA2002/20.pdf.)
17. Boden, M. (1994): Agents and creativity. In Communications of the ACM **37:7** 117–121.
18. Boden, M. (1996): The Philosophy of Artificial Life. Oxford University Press, New York, NY.
19. Dorin, A. (2001): Aesthetic fitness and artificial evolution for the selection of imagery from the mythical infinite library. In Keleman, J. and Sosik, P. (eds.) Advances in Artificial Life - ECAL 2001 Proceedings, Springer-Verlag, Berlin, LNAI 2159, 659–668.
20. Ramachandran, V. and Hirstein, W. (1999): The science of art: a neurological theory of aesthetic experience. In Journal of Consciousness Studies **6** 15–52.
21. Whitelaw, M. (2004): Metacreation: Art and Artificial Life. MIT Press, Cambridge, MA.
22. Harlan, R., Levine, D., and McClarigan, S. (2000): The Khepera robot and kRobot class: a platform for introducing robotics in the undergraduate curriculum. Technical Report 4, Bonaventure Undergraduate Robotics Laboratory, St. Bonaventure University, New York.

Consensual Paintings

Paulo Urbano

Faculdade de Ciências de Lisboa
pub@di.fc.ul.pt

Abstract. Decentralized coordination can be achieved by the emergence of a consensual choice inside a group of simple agents. Work done on emergence of social laws, and on emergence of a shared lexicon, are known examples of possible benefits of consensus formation in multi-agent systems. We think that in the artificial artistic realm, the agreement on some individual choices (attributes, behaviour, etc) can be important for the emergence of interesting patterns. We describe here an effective decentralized mechanism of consensus formation and how we can achieve a random evolution of decentralized consensual choices. Our goal is designing swarm art, exploring the landscape of forms. Non coordinated social behaviour can be unfruitful for the goal of collective artistic creation. On the other hand, full agreement along time generally leads towards too much homogeneity in a collective pattern. This way, the random succession of collective agreements can lead to the emergence of random patterns, somewhere between order and chaos. We show several application of this transition between consensual choices in a group of micro-painters that create random artistic patterns.

1 Introduction

The emphasis of this paper is on the design of micro-painters swarms, which are able to create interesting patterns in artistic terms. There are already examples of collective paintings inspired by social insects: L. Moura [1] has used a small group of robot-painters inspired by ants' behaviour, which move randomly in a limited space. Stimulated by the local perception of the painting they may leave a trace with one of their coloured pens. The painters rely on stigmergic interaction [2] in order to create chaotic patterns with some spots of the same colour. Colour has the pheromone role: a spot dominated by a certain colour has the capacity to stimulate the painter-robot to add some paint of the same colour. Monmarché et al. [3] have also designed groups of painters inspired by ants' pheromone behaviour. It is based on a competition between ants: the virtual artists try to superimpose their colours on traces made by others, creating a dynamic painting that is always changing. His painters have the capability to "sniff" the painted colours on the environment and react appropriately. The societies are composed by a small number of individuals (less than 10). We [4] have made experiments in swarm painting using also ideas of stigmergy, where painters are attracted by the state of the tableau spots (presence or absence of ink).

Achieving consensus by a decentralised mechanism can be very useful for the coordination of a population of autonomous individuals, where the figure of a leader does not exist. It is the case of common lexicon emergence among a population of agents. Kaplan [5] has studied the dynamics of consensus, in the case of research on language, specifically the formation of a shared lexicon inside a population of autonomous agents. In multi-agent AI systems, it can be crucial that agents agree on certain social rules, in order to decrease conflicts among individuals and promote cooperative behaviour. Shoham and Tennenholtz [6] have tested, experimentally and analytically, convergence properties of several behaviours for the on-line emergence of social conventions inside multi-agent systems.

In general, research on convention emergence assumes that all of the options in confront are equivalent in quality terms, and any of them has the same probability to be chosen by all individuals as a convention. In this case, what is important for coordination and for behaviour simplification is that every one agrees on a rule (driving on the right is a human example)—the nature of the rule is not relevant.

Our goal is to make a population make consecutive collective decisions, producing a kind of a random evolution of collective choices. The nature of choices and the duration of each consensus have to be completely random. With this goal in mind, we want to design behaviours that assure fast convergence, inside a population of parsimonious individuals, in situations of maximal competition (worst case). But, at the same time, we want also the possibility of dissidence of one or more agents in situations of unanimity and that one of the dissidents imposes efficiently a new choice to others. This way we will achieve a random evolution of consensus controlled through an auto-organized mechanism.

We have tried, without success, to adapt the existing behaviours to dissidence and consensus evolution. Thus, we introduced a behaviour that demonstrated better results in what concern convergence velocity in convention emergence. This behaviour revealed to be also suited to dissidence and to the random evolution of consensual choices along time.

We thought that one natural application of this dynamic random choice process would be the artistic world. So, we have created a group of micro-painters (Gaugants), which are able to move and paint inside a virtual canvas. The Gaugants are able to adopt consensual decisions, with direct implications on their coordination, creating complex artistic patterns.

In the next section (2) we describe and analyse the most known behaviours for convention emergence, showing their limitations for the goal of cycles of breaking and forming consensus; in section 3 we introduce a new behaviour based on the notion of force and conflict interaction which is able to converge quickly in worst case situations and it can be easily adapted to breaking and forming consensus ; in section 4 we describe in detail, this successful behaviour; in section 5 we apply the random consensual sequence mechanisms to the artistic world. We will incorporate them in a society of micro-painters (swarm-painters), the Gaugants, are able to create consensual sequences around colour and we close the paper by presenting our conclusions. All our Gaugants paintings were made in Starlogo [7].

2 The Emergence of Conventions

We will describe and discuss the main algorithms on convention emergence developed in lexical and social rules formation research. Agents are able to make their own decisions, and they can interact with each other, influencing and being influenced by others. With time, a winning option can eventually emerge and a consensus is therefore attained. The main goal is to reach a global consensual choice through decentralised mechanisms based on self-organisation. In every model each agent has only local access to the society, which is composed of anonymous agents.

2.1 Interaction Games

The interaction games are based on a series of dialogues involving a pair of agents. In each dialogue, two of the society members are randomly chosen for interaction, the hearing and speaking elements. These names were used in the context of language games, and we maintain them in this paper although, hearing and speaking are used here in a metaphorical sense in the course of a unilateral interaction. During an interaction, the hearing agent gets access to the speaking agent state, and can change his option based on the speaking agent information. These games differ in the type of agents' behaviours.

Initial situation and behaviour evaluation. Speaking about performance analysis, we are interested especially in the average convergence velocity and its variation with the number of agents. The convergence velocity is the number of dialogues necessary for reaching a global consensus, starting with options that are equally distributed among agents—no option dominates in the initial population.

In the research of convention emergence, initial situations correspond to situations of maximal competition. In the literature, two initial situations of maximal competition are considered. In the first one we have only two choices where each one is adopted by 50% of the population, and in the second one we have a different choice per individual.

2.2 Behaviours

We are going to describe and discuss the most important algorithms that were designed in the course of convention emergence research

Simple imitation. In the imitation game, agents are defined just by the options they use in order to name a particular object. During a dialogue, the speaking agent indicates to the hearing agent the option it is currently using, and the latter adopts it immediately. Starting with 2 or N options equally distributed in the population (N agents), nothing directs the group towards convergence, as every option can increase its influence with equal probability. In general, convergence is assured after an important series of oscillations in a time quadratic with the number of agents.

Positive reinforcement with score. The strategy that agents use in this game is to adopt the most diffused option they have seen in the population. In this behaviour, positive reinforcement with score (PRS), players associate a score with each option. An agent is defined by his own option and by a preference vector whose dimension is equal to the total number of options present among the population. Agents register on this data structure the number of times they met a member of each option. During a dialogue, the speaker chooses as its current option the one with highest score in its preference vector (in case of a draw, one of the winners is chosen randomly) and subsequently the hearing agent increases by one unit the corresponding counter. Kaplan has studied the dynamics of convergence for both initial situations of maximal competition. Every counter in the preference vector starts with 0 except for the option initially adopted—the respective counter starts with one unity. In this game, convergence is quicker than simple imitation and the variation is n log n.

Positive reinforcement with a forgetting mechanism. Even before the work on language games, Shoham and Tennenholz [6] have made a series of experiments on the emergence of a consensus where the collective choice was a social rule. They have compared experimentally a number of different individual behaviours, where one of them was very similar to Kaplan's PRS referred above. They introduced two different forgetting mechanisms in PRS and one of them happened to improve the social rule emergence efficiency, reducing the average number of interactions necessary to form a collective choice over a number of simulations. The latter forgetting mechanism consists in fact in possessing a small-term memory (length N) instead of a counter, where only the last N choices seen during the last N interactions are registered. The agent only adopts another option if this new one was seen more times than his current option during the last N encounters. Agents began the games with empty memories. In fact, the simple imitation game of Kaplan corresponds to a full forgetting, the short-term memories of the agents have only one cell (they just register the current encounters). We have reproduced the experiments of Shoham and Tennenholtz for games up to 1000 players, for different memory lengths, and we concluded that the most efficient size is 5 and 7 units for 2 and N (one for each individual) initial options.

2.3 The Goal of Repeatedly Breaking and Forming Consensus

We wish a scenario where it is desirable to alternate between different collective options, i.e., a succession of consensual situations. How could we work out a decentralised strategy related to the described convention behaviours in order to invert a consensual situation? We have tried to introduce a dissidence component in the most behaviours described earlier, but finally we found that it is more suited to the decentralized achievement of stable non-changing conventions. Note that we wish not only the success of a consensus inversion, but also the dynamics of an epidemics' diffusion, with no resistance, with focus on the dissident agent.

The natural idea was to make a dissident become stubborn for a while after changing randomly his choice: this way he would maintain his new choice influencing the others with whom he meets. We consider that stubborn state is not accessible to the other players—agents only communicate their options. Compared to the other behaviours, a stubborn agent would be more successful interacting with simple imitators. But, the performance would be very poor. In what concerns consensus inversion by a stubborn agent we concluded that performance decreases as the memory of encounters length is increased. An agent that memorizes the last three or four encounters will be much more difficult to convince than a simple imitator.

This way, the already known behaviours for convention emergence are unsuited for our goals. Even in the case of only one dissident. We did not even treat other essential problems: what happens if we have more than one stubborn agent at the same time, adopting different options? And what is the appropriate moment to end the temporary stubborn behaviour as we want a succession of several consensual choices?

3 Conflict Interactions for Consensus

In our effort to look for other behaviours suited for our goals, we have introduced force as a new attribute of agents, besides choice. The force attribute transforms each dialogue in a conflict interaction. The main point of conflict interactions is that agents only imitate other agents stronger than them.

3.1 Double Imitation of Stronger Agents with Reinforcement

In this successful behaviour, during encounters, the loosing agents imitate both the force and the option of the winning adversaries. The new recruited agents will have their force increased having more power to recruit. At the same time, equals reinforce their force, independently of loosing or winning. This way, we add a positive reinforcement if they have the same options. Let's look at the behaviour in more detail.

If the speaking agent is stronger (or have identical force) than the hearing agent, the latter will adopt the choice and the force of the former. If the speaking agent is weaker than the hearing agent, this one will conserve both force and option. In case they have the same option at the beginning of interaction, force is reinforced in one unit, independently of loosing or winning. In sum, the stronger ones recruit weaker agents (these will be at least as strong as the winners imitating their choices, and they can even overpass them in case their options were the same) enlarging the influence of their options.

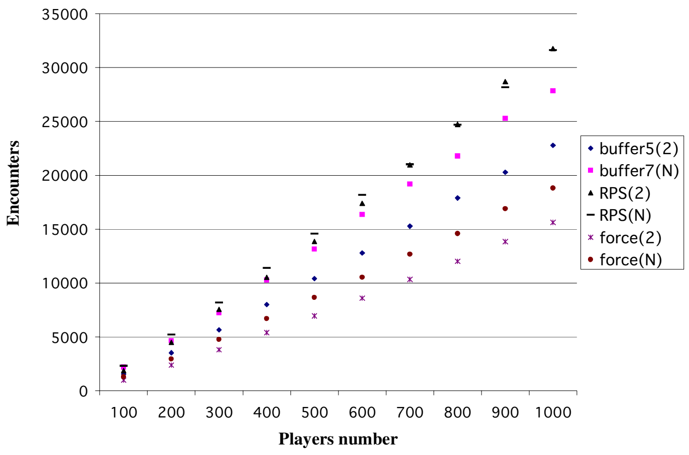


Fig. 1. The difference of performance of PRS, PRS with short-term memory and our behaviour: imitate the stronger

We have conducted a series of experiences [8] in order to test the efficiency convergence until consensus in situations of maximum competition (equal initial force and equally distributed choices). Our experiments showed convergence for populations up to 1000 elements, with 2 and N initial options (N = number of players) equally distributed, where each players starts with force of 0 units. The velocity of convergence is faster than in the PRS game with the forgetting mechanism. Figure 1 compares both games, in the case of 2 options and N options, plotting the average number of encounters necessary for consensus on 1000 simulations for a population varying from 100 to 1000 players.

3.2 Adaptation to Dissidence

We have verified experimentally that forces tend to be relatively homogeneous after consensus. This way, the force of an agent can serve as a local measure of the others' forces. If a solitary representative of an option has significantly more force than all the other agents (they all belong to the opposition), than his option will always dominate the whole population with the efficiency of a diffusion process with origin in one agent. We have tested populations varying from 3 to 1000 elements, where the solitary agent starts always with 200 times more force than the others, and we have always found a successful consensual inversion. Thus, to reverse consensual situations we just need that an agent changes to another option and increases his force by a significant value (for example, 200).

Therefore, our behaviour showed good results in what concerns fast convergence in situations of maximum competition between choices and also to the capacity of a stronger individual alone to influence the whole population reversing a consensual situation and achieving a new consensus. Here, force functions as kind of power to influence and recruit others.

4 Random Consensus Succession

We are now ready to discuss the agent behaviour suited for achieving a random collective succession of consensual choices. In principle, dissidence must appear during a consensual situation. But how can an agent, constrained to have local access to others, knows that everybody adopted the same option? He just can't. But he can count the number of consecutive equals he encounters (agents with the same choice as his own)

4.1 Memory of the Number of Consecutive Equals

Each agent will possess a counter where he registers the number of consecutive agents he meets with the same option after he has adopted his current one. So whenever he meets an "equal" he increases his counter and whenever he meets a "different" he resets his counter to zero. Thus, this memory is a local measure of the choices adopted by the whole group.

4.2 Dissidence Threshold and Probability

Now we will answer the question: But when does an agent adopt the dissident behaviour? After consecutively seeing some reasonable number of agents with the same option. This number is called the dissidence threshold and is a local attribute. After having seen at least a certain number of consecutive equals an agent will be a dissident with some probability, which is called the dissidence probability. The most important parameter is the dissidence threshold. In fact, consensus duration depends very much on this parameter. This way, every agent should adopt the same dissidence threshold, and it should also evolve in order for the emergence of different durations of consensual periods.

4.3 The Final Behaviour

Now we are ready to give in detail the final behaviour. Each agent possesses 4 attributes: choice, force, dissidence threshold *(dt)* and probability *(dp)*. During a random encounter two agents face each other and the speaking agent reveals his choice and his force. The hearing agent updates his equals' counter. If his counter is less or equal to *dt* than he becomes a dissident with probability *dp*. To be a dissident is to increase his force in 200 units and to change the option to another, chosen randomly; he also chooses randomly a new dissidence threshold. After this stage the agent will fight in his head with his partner. If he is not stronger than him, he will imitate both his force and his choice, otherwise he conserves both. After the fight, comes the reinforcement: if he had the same choice as his partner when they met than his force is reinforced (one more unit), independently of the fight outcome.

We know that our behaviour is maximally efficient in situations of just one dissident. A bigger number of dissidents implies, generally, larger transition periods. The exception is the case we have more than one dissident with the same choice—the efficiency will obviously increase. In the worst case, every individual has a different choice and we know the performance is good, even in the unlikely situation where they have the same force (better than the standard behaviours, see fig. 1).

In each consensual period, every individual will have the same option and the same *dt*. In each dissidence point, those two attribute values will change randomly, which means that we'll have a random consensus evolution concerning the nature and duration of each consensus. Through the variation of both parameters, we can obtain a large diversity of consensual periods and transition periods between two consensual choices. We want, in general, short transition periods and variable consensus durations, constraining these two parameters *dt* e *dp*. Note that there are cases where consensus is not even achieved-just think of a zero *dt* or even a very small value.

5 Random Consensual Paintings

The Gaugants are a swarm of small artificial micro-painters, which are able to paint a bi-dimensional (toroidal) virtual canvas, composed of small cells. Each patch possesses an attribute: colour. There is a fixed colour (usually grey) for the background. Any other colour corresponds to paint. Initially, we launch these painters in a non-painted background, each one occupying a particular cell, and they will move along, depositing a trace of ink, until the canvas is completely fulfilled. Note that each painter is constrained to paint only non-painted cells and when there isn't any non-painted cell left, the artistic work cannot change and is considered finished. We can also stop the painting after some pre-defined time. After setting up the society, we have a sequence of steps where in each time step every Gaugant executes its own behaviour until the painting is finished.

5.1 The Gaugants

Each Gaugant is very simple and has a very simple social behaviour. Each one has a position (real Cartesian coordinates), an orientation (0..360), and can only inhabit one cell, the one that corresponds to the round of their coordinates. They have a certain speed, which is a global parameter—speed corresponds to the number of cells they move forward in each step. On the other hand, the painters are created with a particular colour. They can interact unilaterally with any other painter independently of their position in order to exchange information unless they are constrained by some radius of perception. Our micro-painters will have incorporated the imitation and dissidence behaviours we described above, based on force.

5.2 Consensual Colours

The micro-painter agents will establish a consensus around colour. Here is the general behaviour:

1. If counter number is no less than the dissidence threshold, change to a dissident with probability dp (if success go to 2 else to 3).
2. Turn into a dissident: mutate and increase force in 200 units; choose a random dissidence threshold and reset the counter of equals.
3. Choose a random partner. If his partner is stronger then imitate him, otherwise does not change. Gaugants always imitate the force and equals-threshold of stronger agents, during interactions—it is essential for our consensual sequence formation. They also imitate colour because this is the focus of consensus. But they can imitate other attributes as well, that we find important for pattern creation.
4. If he had the same colour as his partner at the beginning of their encounter, then increase force in 1 unit (reinforcement). Update the equal's counter dependent on the colour of his partner (1 unit more in case they were the same and 0 in case they were different).
5. Try to paint his patch (only if it is not yet painted).
6. Executes its own non-interactive behaviour (normally related with movement).

Mutation involves always colour: the dissident changes to a different colour (normally randomly chosen) and he changes also the equals-threshold. However the dissident may also change other parameters.

5.3 Experiment 1

Besides position, orientation, colour, force and equals-threshold, each Gaugant has a parameter called Rot In the beginning we divide the global population in a number of groups and put each group with the same orientation and position. The agents choose their colours in a random fashion. The same happens with the Rot parameter. There is a global parameter MaxRot and each agent sets his Rot to a random value inside the interval [0,MaxRot]. So, in the initial situation, inside a group, the painters have the same position and orientation, but different colours and Rots. The Gaugants only imitate the colours of the others (Rot is fixed since the beginning). In this experiment we consider that each painter may communicate with any other. The non-interactive individual behaviour consists only in rotating Rot degrees and going forward a number of steps (speed). Speed is a global parameter and also not subject of imitation. The dissident only changes his colour (mutation).

Fig. 2 shows three snapshots of a painting evolution along time (population of 2000 painters divide in groups of 20 groups 100 elements.

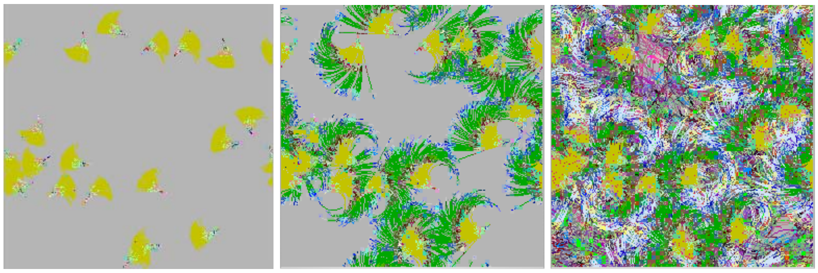


Fig. 2. Evolution of a painting. 2000 micro-painters divided in 20 groups of 100 elements, dp = 0,001. MaxRot is 20 and any agent can imitate another (global communication).

We can see that initially every group element is in the same patch but because they do different rotations, the one-colour spots get larger and larger, and after a while, every agent is dispersed in the tableau creating a confused background that highlights the initial spots. The fact that any micro-painter can choose any other as a partner is the responsible for having similar forms and colours in different parts of the tableau. We can see also different consensual areas, implying different consensus durations—this is due to the change of the equals-threshold during dissidence.

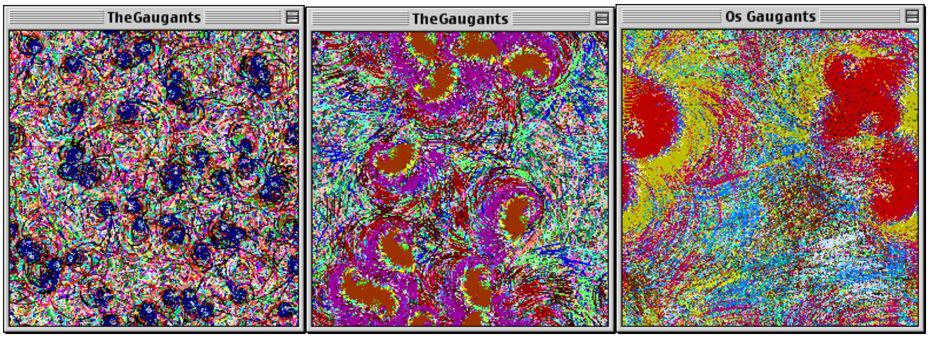


Fig. 3. Gallery of 3 Gaugant paintings

In figure 3 we show 3 paintings that were made by different population dimensions, initial group divisions, speed and MaxRot. For every one we have a population of 2000 agents and dp=0,001. These and other pictures can be seen in http://www.di.fc.ul.pt/~pub/gaugants.

5.4 Experiment 2

In the second experiment Gaugants will imitate both colour and orientation. The dissident will change colour and orientation. Moving is just going forward a number of speed units (global parameter) and rotating to the right a random number of units (between 0 and Rot). MaxRot is 6 and each group imposes to theirs elements the same position and orientation. Each element begins with a random colour and Rot is randomly chosen between 0 and MaxRot.

In figure 4 we show the evolution of a painting made by a population of 2000 elements divided in 30 groups where everybody can choose any other as a partner to interact. We can see clearly the sequence of consensus (specially due to colour, the change of orientation was very light).

Looking at figure 5, showing a painting that we call *The Swans* we can see clearly that orientation is changed during dissident behaviour.

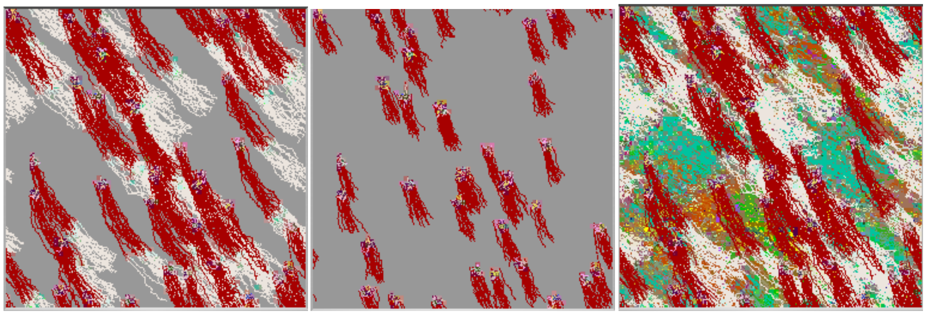


Fig. 4. Three snapshots of a painting by 2000 agents that imitate both colour and orientation

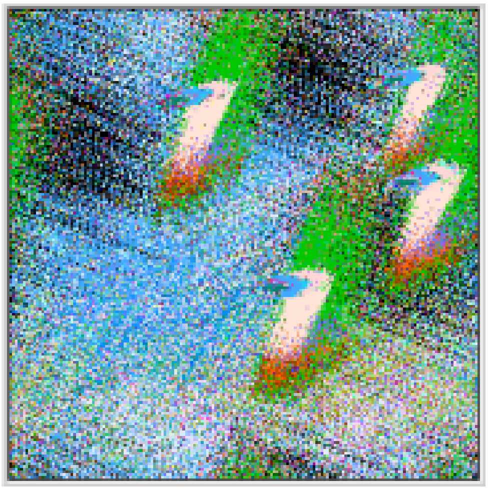


Fig. 5. The swans. Four initial groups of 500 agents each.

6 Conclusion and Future Work

Achieving consensus in a population of autonomous agents can be very useful for the co-ordination of a swarm of simple entities. Following work on imitation in the area of convention emergence, we have developed a new model for achieving consensus. Our model uses a force attribute and behaviour is based on imitation. During a conflict interaction, when an agent looses the combat it will imitate the option of the winner. At the same time the looser will be a little stronger than the winner. We are in presence of a process of recruitment where the strong recruits the weak. We have analysed this algorithm through a series of experiments and concluded that it has good properties, namely fast convergence towards sharing a global option and the capacity to select collectively the best quality options, being well adapted to a changing world. More, it is well suited to dynamic optimization, where the value of options can change based on the context.

We have applied this collective mechanism of emergence of random consensual sequences to the artistic world, namely to the production of collective paintings. We introduce the Gaugants, a society of micro-painters and we show some examples of paintings somewhere between order and chaos. Consensus around some attributes can be the source of some order and pattern but breaking consensus and new consensus formation can be the source of diversity and non-homogeneity.

In the Gaugants artistic world, we are also extending imitation beyond certain local parameters (the case of colour or orientation). Our aim is that not only attributes but also behaviours are subject to imitation creating more variations in the random-patterns made by a swarm of micro-artists.

References

1. Moura, L.: Swarm Paintings. Architopia: Art, Architecture, Science. (ed. Maubant) Institut d'Art Contemporaine (2002)
2. Grassé, P.-P.: "Termitologia, Tome II." Fondation des Sociétés. Construction. Paris: Masson, (1984)
3. Aupetit, S., Bordeau, V., Monmarché, N., Slimane, Mohamed, Venturini, G. "Interactive Evolution of Ant Paintings", in CEC´03 - Congress on Evolutionary Computation, IEEE Press, Canberra, Australia, (2003) 1376,1382
4. Urbano, P.: Playing in the Pheromone Playground: Experiences in Swarm Painting. In: (proc. EvoMUSART'05). Applications of Evolutionary Computing, EvoWorkshops 2005. Lecture Notes in Computer Science. Volume 3449., Springer- Verlag (2005), 527-532
5. Kaplan, F.: L"emergence d'un lexique dans une population d'agents autonomes. PhD thesis, LIP6 Universit'e Paris VI, (2000).
6. Shoham, Y, Tennenholtz, M.: Emergent conventions in multi-agents systems: initial experiments results and observations. In Proceedings of the 3rd International Conference on Principles of Knowledge and Reasoning, (1992) 225-231.
7. Resnick, M.: Turtles, Termites and Traffic Jams: explorations in massively parallel microworlds. MIT Press (1994).
8. Urbano, P. (2004)–Jogos Descentralizados de Consenso ou de Consenso em Consenso. PhD Thesis, Universidade de Lisboa, (2004).

Using Physiological Signals to Evolve Art

Tristan Basa, Christian Anthony Go, Kil-Sang Yoo, and Won-Hyung Lee

Graduate School of Advanced Imaging Science, Multimedia and Film,
Department of Image Engineering, Chung-Ang University,
Seoul 156-756, Korea
trisb@hotmail.com, chipgo@gmail.com, lucky@wm.cau.ac.kr, whlee@cau.ac.kr

Abstract. Human subjectivity have always posed a problem when it comes to judging designs. The line that divides what is interesting or not is blurred by the different interpretations as varied as the individuals themselves. Some approaches have made use of novelty in determining interestingness. However, computational measures of novelty such as the Euclidean distance are mere approximations to what the human brain finds interesting. In this paper, we explore the possibility of determining interestingness in a more direct method by using learning techniques such as Support Vector Machines to identify emotions from physiological signals, and then use genetic algorithms to evolve artworks that resulted in positive emotional signals.

1 Introduction

Even today, automation has only been effectively applied on systems which are mechanical in nature. Even with a specialized field like artificial intelligence, activities which involve human creativity continue to present a major challenge as far as automation is concerned. Examples of these are systems requiring human preference. In producing designs amiable to humans, a system must be able to model what humans find interesting. A system which is able to recognize the interestingness of something will also be able to mimic curiosity. One major issue in building such systems is that human preference tends to be subjective. While one person's choice might not be the same as that of another, several theories have been proposed which suggest a common psychological pattern involved in the process of making those choices.

One approach that has been explored is the use of novelty to calculate interest[1]. Novelty can be calculated using machine learning techniques such as neural networks, but in this context, these techniques can be viewed as mere approximations to the way the human brain recognizes novelty, and based on our experiments, there are cases when interest is not necessarily a function of novelty. On the other hand, studies have been done which suggest basic emotions having physiological signatures[2]. These patterns can be used as hypotheses with which to detect interest. This approach, intuitively, is a more direct measure of interest.

Although the use of physiological signals has been used directly in producing art[3], Our application involves determining interesting two-dimensional digital artworks which can be used as parents to evolve succeeding generations of art using genetic algorithm. Other possible applications of this technique can be in any other forms of art which elicits physiological reactions. Another application could be in the field of medicine. This technique can be used to help quadriplegic people communicate emotions.

In the following section, we discuss the use of genetic algorithms in evolutionary art. In section 3, we introduce Support Vector Machines. In Section 4, we discuss the methodology and experimental setup. Section 5 presents the results of the experiments. The last section discusses implications of this research and future directions.

2 Genetic Algorithms and Art

Genetic algorithms (GAs) are learning methods motivated by the biological evolutionary process. GAs generate succeeding hypotheses by repeatedly combining and mutating the best ones based on a fitness function[4]. They are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination or crossover. Genetic algorithms are typically implemented as a simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimization problem evolves toward better solutions. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm.

Artwork can likewise be evolved using genetic algorithms [5]. The basic idea behind creating art from mathematical equations is to produce a color for each pixel from a formula operating on an x,y coordinate. Normally, the artwork is steered by a human operator who selects members of a population based upon some aes-

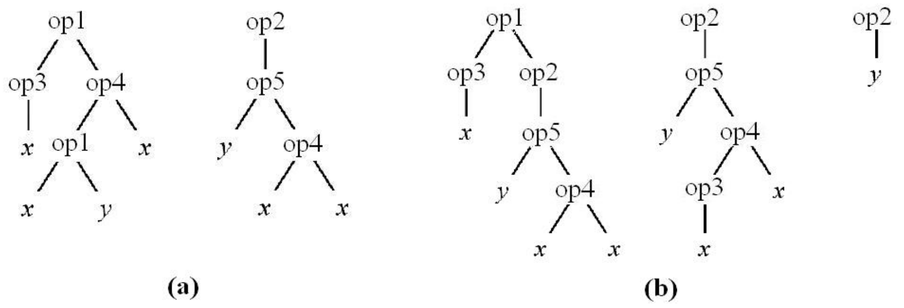


Fig. 1. (a) Two parent trees that can be used to generate artworks. (b) Three possible children of both parents.

thetic criteria. Using genetic algorithms, we can form new formulas from the previous formulas via the “crossover” and the “mutation” operations. The crossover operator exchanges a randomly chosen branch of one parent tree with a randomly chosen branch of the other parent to generate children as illustrated in Fig. 1.

3 Support Vector Machines

Support Vector Machines (SVMs) are learning systems that use a hypothesis space of learning functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory[6]. When used for classification, the SVM algorithm attempts to create a hyperplane that separates the data into two classes with the maximum-margin, called an optimal separating hyperplane (OSH). Given training examples labelled either “yes/active” or “no/inactive”, a maximum-margin hyperplane is identified which splits the “yes/active” from the “no/inactive” training examples, such that the distance between the hyperplane and the closest examples -the margin, is maximized. Figure 2 illustrates a simple example of how an OSH divides two dimensional data. The use of the maximum-margin hyperplane is motivated by Vapnik Chervonenkis Theory, which provides a probabilistic test error bound that is minimized when the margin is maximized. However the utility of this theoretical analysis is sometimes questioned given the large slack associated with these bounds: the bounds often predict more than 100% error rates. The parameters of the maximum-margin hyperplane are derived by solving a Quadratic Programming (QP) optimization problem. There exist several specialized algorithms for quickly solving the QP problem that arises from SVMs.

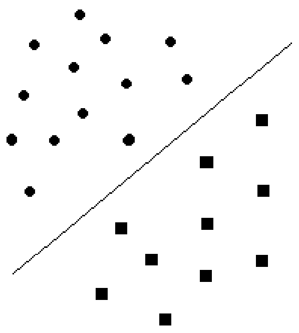


Fig. 2. An Optimal Separating Hyperplane divides the data into two classes

4 Methods

Human Electroencephalography (EEG) measures both the frequency and amplitude of electrical activity generated from the brain. The growing use of EEG has enabled researchers to study regional brain activity and brain function, in particular, emotional activity. In this paper, we basically want to distinguish positive emotions from negative ones. We use the basic emotions based on the Facial Coding System (FACS)[7] as our positive and negative examples of emotions because of their universality and assumed innate neural substrates. The FACS basic emotions are classified as: Anger, Sadness, Happiness, Surprise, Disgust and Fear.

Levenson[8] demonstrated that voluntary facial activity produced significant levels of subjective experience of a certain emotion. Autonomic distinctions were identified between positive and negative emotions. This also extended to include distinctions between some negative emotions. Recorded heart rate was also found to differentiate emotions.

Hubert and De-Jong [9] showed different electrodermal responses and heart rates when subjects were exposed to film stimuli to elicit a positive and negative response. A temporary decrease in heart rate was observed during negative stimulation, while it remained unchanged during positive stimulation. Skin conductance increased significantly during negative stimulation and it recorded only an initial increase during positive stimulation. Experiments by Collet demonstrate that each basic emotion has a specific Autonomic Nervous System (ANS) response pattern associated with it. Fifteen out of fifteen emotion pairs were distinguished using combined electrodermal (skin resistance, skin conductance and skin potential), thermo-circulatory (skin blood flow and skin temperature) and respiratory parameters (instantaneous respiratory frequency). Emotions were redundantly separated thus supporting the hypothesis of ANS specificity[2].

It has thus been proven that there exists specific Autonomic Nervous System parameters which can be associated with each basic emotion. These parameters will serve as our basis of the genetic algorithm for selecting which artwork to evolve.

Due to the stochastic nature of EEG signals, patterns specific to an emotion are not identical. As such, machine learning techniques can be employed to infer models of human emotions from these signals.

Support Vector Machines have been shown to be excellent in inferring boolean functions from a training set of positive and negative examples. In our experimental setup, the positive examples refer to emotions such as happiness, excitement, and surprise, and the negative examples to emotions such as sadness, disappointment, and anger.

By means of using an image exposure technique, different emotional states were elicited from the test subjects. The subjects consisted of 8 volunteers (4 male and 4 female; between 21 and 32 years of age). All of the subjects in this experiment were graduate students whose visual art backgrounds range from completely none to 3D graphic designers. They gave informed written consent to the study and were not paid for their participation. All were in healthy condition and did not take any prescribed medication.

An image pool of 72 images consisting of 36 positive (happy) and 36 negative (sad/depressing) was compiled. These images have been selected at the discretion of the facilitator. Each participant was asked to select 2 images from the pool that stimulated the strongest positive and negative response. Following an initial resting state of two minutes to calibrate the EEG machine per participant, subjects were shown their respective selected images. Each subject was shown their respective image to elicit the corresponding emotion, followed by a 30 second resting period in between emotions. The emotional mood states were highly intensive and maintained for at least one minute. This is a period sufficiently long for valid estimates of EEG activity to be measured.

Subjects were required to refrain from smoking and consuming caffeine and stimulants 2 hours immediately preceding the experiment to prevent irregularities in ANS parameters. Absolute silence was observed during all experiments to prevent signal artifacts.

The ANS parameters recorded were respiration, ECG and 8 channels per frequency, Alpha, Theta and Beta. Alpha (Berger's wave) is the frequency range from 8.5 Hz to 12 Hz. It is characteristic of a relaxed, alert state of consciousness and is present by the age of two years. Beta is the frequency range above 12 Hz. Low amplitude beta with multiple and varying frequencies is often associated with active, busy or anxious thinking and active concentration. Theta is the frequency range from 4.5 Hz to 8 Hz and is associated with drowsiness, childhood, adolescence and young adulthood. This EEG frequency can sometimes be produced by hyperventilation. Theta waves can be seen during hypnagogic states such as trances, hypnosis, deep daydreams, lucid dreaming and light sleep and the preconscious state just upon waking, and just before falling asleep. It was observed that the women in the group recorded a more pronounced difference between emotional states compared to the men. The following experimental results will demonstrate the effects of better training data with regards to emotion detection.

The collected EEG test data were then classified into training models used for the learning algorithms of Support Vector Machines. SVM were trained to recognize positive and negative emotions using these EEG models. 16 sets of training data were utilized, comprised of 2 sets of emotions (positive and negative) for each of the 8 participants. Upon completion of SVM training, generalized templates of positive and negative emotions were created. These emotional templates were the basis upon which the SVM would compare and classify EEG test data, as either a positive emotion or a negative emotion (Fig. 3).

During the testing phase, each participant was asked to sit in a comfortable armchair and connected to the EEG machine. The same restrictions and controlled conditions applied to the test subjects, no stimulants, no smoking and absolute silence during the experiment. Digital art images were presented in a monitor in front of each subject for one minute while EEG signals were being recorded. The sequence of presentation of the images was randomly selected. The group of initial images all belonged to a single "family", they were all evolved from the same parents. Using SVM classifier, the artwork that resulted in the most positive classification was selected to be the image to be evolved. A negative response to the artwork signals disinterest/boredom, prompting the image to be

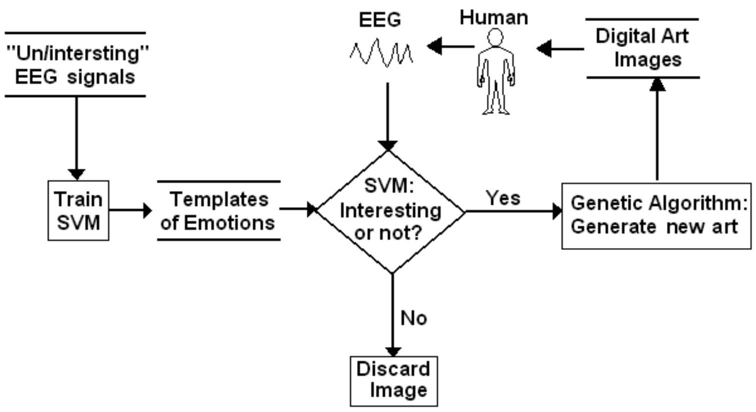


Fig. 3. Flow diagram of using physiological signals to evolve artworks

discarded. A 30 second resting period was observed in between images. During this resting period, a blank white screen is showed to the participants. Evolution of artwork was carried out until the third generation of offspring. Each selection made by SVM was recorded for post-experiment evaluation. In order to verify the correctness of the automatically chosen artwork, the subjects were exposed to the same images immediately after the experiment and asked to recall their choice. This was validated against the record of interesting artworks determined by SVM.

5 Results

In the testing phase, six subjects participated out of the eight who volunteered in the training phase. SVM was able to classify eleven out of eighteen actual data correctly from the six subjects. This translates into a 61% accuracy. Consistent with the training data, it was also observed that the females in the group recorded a higher percentage of correct emotions determined (55%) as compared to the males (45%). This could be attributed to more consistent signals extracted from female subjects. Sample images used in the experiment are shown in Figure 4. Figure 5 shows the accuracy obtained by SVM classification. Figure 6 illustrates the distribution of accuracy between the genders.

6 Conclusion

Previous basis of finding interesting designs have focused on novelty to evolve art. Although novelty has been shown to coincide with human preference to some accuracy, it can be considered indirect measures of what the human brain itself finds interesting. This paper has presented a more direct approach of measuring interest by using physiological signals which can be used as fitness function to evolve new designs.

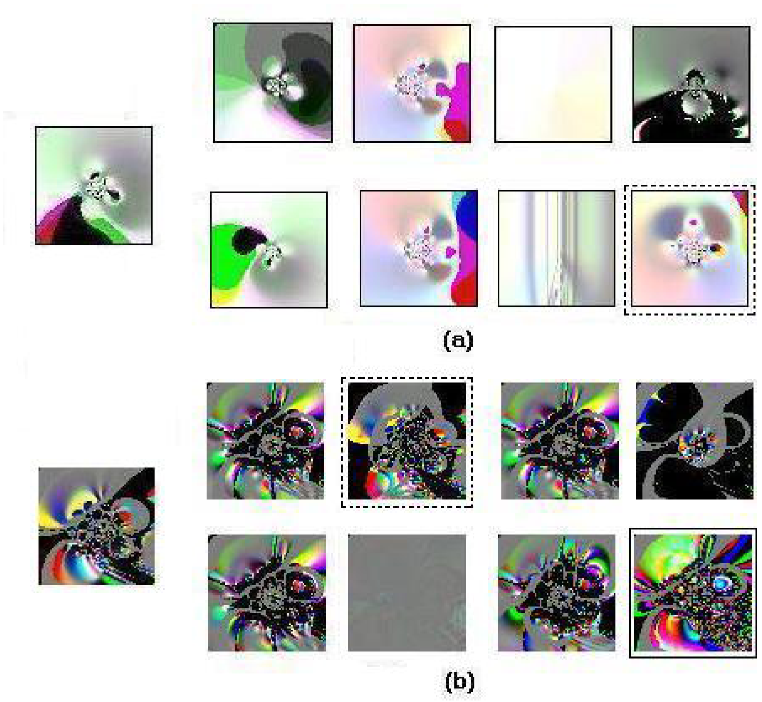


Fig. 4. Genetic art images on the left are the original images. Images on the right corresponds to eight children produced from the parent. (a) An example where the positive classification by SVM matched human preference, enclosed by dotted lines. (b)An example wherein the positive classification by SVM, enclosed by dotted lines, did not match the preference of a subject, enclosed by solid line.

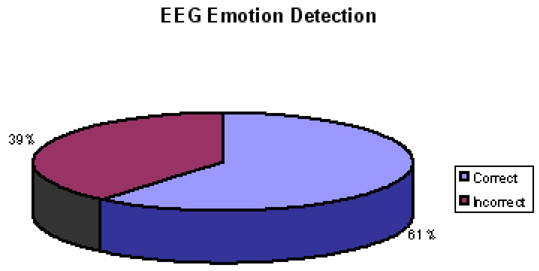


Fig. 5. Accuracy of classification

Experimental results have shown that using machine learning techniques such as Support Vector Machines, human preference in artworks can be generally inferred. For our purposes, we have used this approach to evolve art, however, there are many other facets wherein this technique can also be applied. Other potential areas of application can include the field of medicine. One advantage born of

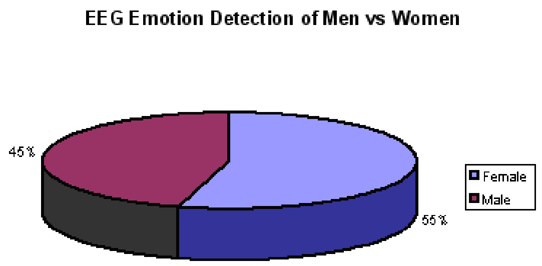


Fig. 6. Distribution of accuracy between male and female subjects

this approach is the possibility of being able to communicate the preferences and emotions of quadriplegic people. However, present methods of extracting physiological signals are still considered cumbersome inasmuch as they still require cumbersome machines.

Future directions of research can extend this approach by incorporating a preprocessing step such as the Blind Source Separation (BSS) algorithm to minimize noise in EEG signals which should improve the classification accuracy.

Acknowledgment

This research was supported by the Korea Culture and Content Agency, Seoul, Korea, under the 2005 CT project and research fund of Chung-Ang University in Seoul.

References

1. Saunders, R.: Curious Design Agents and Artificial Creativity. Proceedings of the 4th conference on Creativity & cognition. Loughborough, UK. (2002) 80-87.
2. Collete, C., Vernet-Maury, E., Delhomme, G., Dittmar, A.: Autonomic Nervous System Response Patterns Specificity to Basic Emotions. Journal of the Autonomic Nervous System 62 (1997) 45-57
3. Mori, M.:Wave UFO: http://www.publicartfund.org/pafweb/projects/03/mori_release_s03.html
4. Mitchell, T.: Machine Learning. McGraw-Hill Companies Inc. Singapore. 1997
5. Sims, K.: Artificial Evolution for Computer Graphics: Computer Graphics (Siggraph '91 proceedings), Vol.25, No.4, July 1991, pp.319-328.
6. Christianini, N., Taylor, J.S.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press. UK. 2000
7. Ekman, P. and Friesen, W.V.: The Facial Action Coding System. Consulting Psychologists Press, Paolo Alto, 1978
8. Levenson, R.W., Ekman, P. and Friesen, W.V.: Voluntary facial action generates emotions specific autonomous nervous system activity. Psychophysiol., 21, 1990
9. Hubert,W. and De Jong Meyer, R.: Psychophysiological response patterns to positive and negative film stimuli. Biol. Psychol., 1990, 73-93

10. Hinrich, H., Machleidt, W.: Basic Emotions Reflected in EEG-coherences. International Journal of Phsychophysiology 13 (1992) 225-232.
11. Fridlung, A.J., Schwartz, G.E. and Fowler, S.C.: Pattern Recognition of Self-Reported Emotional state from multiple-site facial EMG activity during affective imagery. Psyhophysiol., 21, 1984

Science of Networks and Music: A New Approach on Musical Analysis and Creation

Gianfranco Campolongo[1] and Stefano Vena[2]

[1] Department of Linguistics, University of Calabria, Cubo 17b Via P. Bucci, Arcavacata di Rende(CS) 87036, Italy
g.campolongo@unical.it

[2] Department of Mathematics, University of Calabria, Cubo 30b Via P. Bucci, Arcavacata di Rende(CS) 87036, Italy
stefano.vena@gmail.com

Abstract. Science of Networks is a very young discipline whose results have rapidly influenced many different fields of scientific research. In this paper we present some experimentations of a new approach on generative music based on small-world networks. The basic idea of this work is that network can be a useful instrument for musical modeling, analysis and creation. We studied over 100 musical compositions of different genres (classical, pop, rock) by means of science of networks, then used this data for generating algorithms for musical creation and author attribution. The first step of this work is the implementation of a software that allows to represent and analyse musical compositions, then we developed a genetic algorithm for the production of networks with particular features. These networks are finally used for the generation of self-organized melodies and scales.

1 Introduction

In 1998, an important paper [1] demonstrated that the connections between peoples all over the world may be studied as a graph which is not completely random or regular but an ordered lattice with a small quantity of disorder. For the construction of this model Watts and Strogatz used a procedure called "rewiring", which consists of attaching to each edge of a regular lattice, of the probability p that the edge be moved to another vertex. Such probability p, comprised between 0 and 1, determines the randomness of the structure; a graph with $p = 0$ is completely regular and a graph with $p = 1$ is entirely random, for $0 < p < 1$ we obtain a network called small-world. This model is based upon two parameters, path length (L) and clustering coefficient (C). The first parameter measures the average separation between two vertices in a graph composed by N vertices and E edges, the second one is the ratio between the edges present in the neighborhood of a vertex i (which is composed of all the vertices directly connected to the vertex i) and the maximum possible number of edges of the neighborhood, that is given by the formula:

$$\frac{(N_i\,(N_i - 1))}{2} \tag{1}$$

where N_i is the number of nodes of the subgraph of the neighborhood of i. So the clustering coefficient C_i is will be calculated as follows:

$$C_i = \frac{2 * E_i}{(N_i\,(N_i - 1))} \tag{2}$$

Where E_i is the number of edges of the neighborhood of i. The clustering coefficient of the whole network is the average value of all C_i's. Usually, a random graph is characterized by low values of C and L. This means that each vertex communicates with all the others in a small number of paths, but also means that the structure is not enough connected, unlike real networks, in which every vertex (people) results to be more connected with near vertices. On the contrary a regular network has a high clustering coefficient but also a high average separation between vertices. The model called small-world, maintains a good clustering coefficient like regular networks and a very small number of passages are necessary to connect the vertices of the structure. The work of Watts and Strogatz gave a scientific explanation of the phenomenon popularly known as "six degrees of separation", and gave birth to a series of studies which influenced many other fields of scientific research.

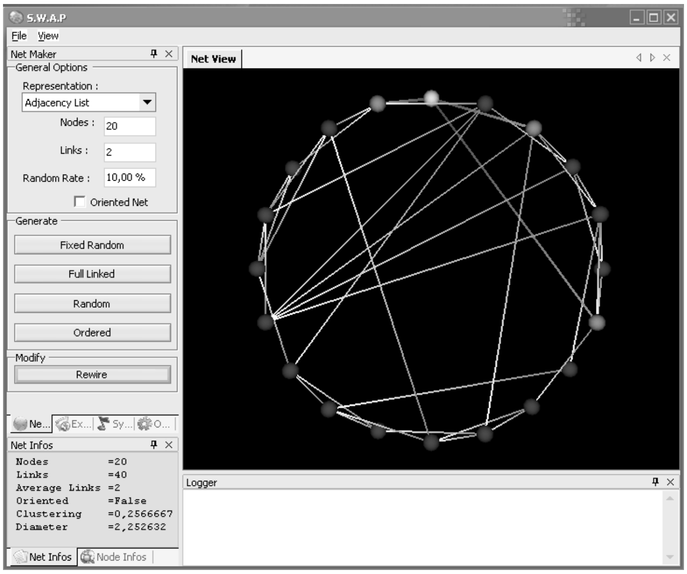
An alternative model of small-world network was given by Newman [2,3,4] which created a graph where the randomness was not introduced by a rewiring procedure but adding to a regular structure a small number of nodes with a high number of links per node. These vertices have the same function of the short paths of the Watts and Strogatz model, since they diminish the average value of L without the risk of disaggregation for some areas of the graph, as it happens in some cases with the rewiring procedure.

Another important contribute to science of networks is given by Albert-Làszlò Barabàsi and his team [5,6,7,8,9]. Their work is particularly focused on "scale-free" networks, like the Internet and the World Wide Web. These networks are characterized by the presence of some hyper-connected nodes, also called hubs, and a large quantity of other nodes with a very small number of links. The work of Barabàsi et al. has shown the most important features of these networks, such as the power law distribution of links in the structure, the presence of different areas (called continents by Barabàsi) in the WWW, and shown the error and attack tolerance of scale free networks. Barabàsi has also studied the average separation between web pages (approximately estimated in 19 passages, or clicks), finding surprising similarities between scale-free networks and the small-world model defined by Watts and Strogatz.

2 Musical Modeling Through Networks

According to Latora et al. [10,11,12] every complex system can be seen as a network where the single elements are represented by nodes, and the links show the interaction between parts. Music may also be thought as a complex system [13], with a strong interaction and emerging properties. Recently, network has been adopted as a metaphor for algorithmic composition and for musical analysis. In some cases [14] the “interconnected musical networks” have been used as instruments to improve interaction in musical performances and as extension of Cage’s work on indeterminacy [15]. Composers such as Weinberg use network as a tool to connect musicians and performers producing brand new interactions and very sophisticated compositional practices (that in some cases involve a great amount of people at great distances). In other cases [16, 17, 18] network becomes an instrument for graphical and analytical modeling of music and a new paradigm for generative and evolutionary music [19, 20].

Every musical composition may be modeled as a network where each node corresponds to a note and each link represents a connection between notes. For instance, if a song starts with F and the second note is A# we will have two nodes connected by a link and so on, till the end of the song. Network becomes a sort of map of the composition, a catalogue of the paths performed by the musician, represented as they were the streets of a city, or the traffic of an airport. All the connections in the map represent melodic intervals of the studied song. This kind of representation is advantageous because it shows, in a single image, the complex plot of interplay in a composition. The connections between nodes clearly show how the notes are linked, so that it is easy to understand how a melody is organized only by watching the graphical structure. We may think network as an instrument for didactic use too. As shown in Fig. 1 a simple picture can help a non expert listener or musician comprehend musical concepts.



Representation : Adjacency List
Nodes : 20
Links : 2
Random Rate : 10,00 %
☐ Oriented Net

```
Nodes          =20
Links          =40
Average Links  =2
Oriented       =False
Clustering     =0,2566667
Diameter       =2,252632
```

Fig. 1. Screenshot of SWAP

Musical structures contained in compositions may be easily characterized; for instance, all melodic intervals or the notes directly connected to a generic note are immediately apparent. In the next section a software for musical modeling and creation by networks will be described.

3 SWAP (Small World Analyser and Player)

SWAP is a software for musical representation, analysis and synthesis. SWAP's interface is divided in three main areas:

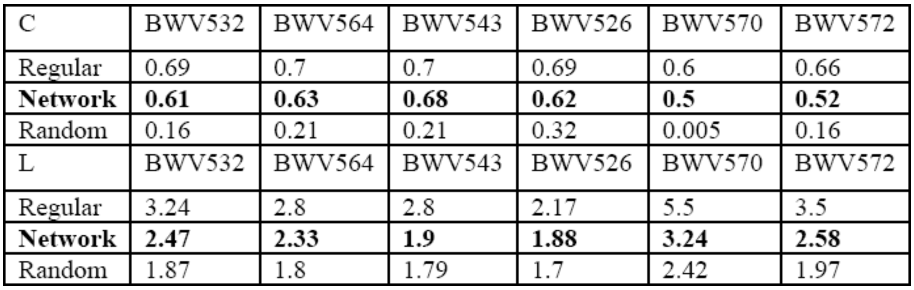
1. Control Panel,
2. Net View,
3. Logger Window.

The Control Panel, situated at the left of the interface, is the area where all the controls, buttons and information are placed. Through this area the user may execute all the operations available in this software. The Net View, at the center of the interface, is the part of the interface where the networks are represented, while the logger window (placed under the net view window) shows all the operations made by SWAP. We may divide the functionalities of SWAP in two parts. The first is representation and modeling function. Giving a MIDI file as input, SWAP creates a representation of the composition as a network. All numerical information on networks can be controlled by the Net Info window placed in the control panel. This window shows all data extracted from the composition, such as number of nodes, edges, average edges per node number, clustering coefficient, path length, and so on. SWAP also contains a window called Node Info, which gives all the information about every single node of the network. The user may select a node directly from the Net View with a mouse click. The selected node will be automatically evidenced on the graph and all the edges connected to this node will change color from white to fluorescent green, so all the notes directly connected to the selected one will be immediately apparent. On the Node Info window will appear all the information on the node: value (A, A#, B etc.), number of links and so on. Another important option is the Mouse Explore function, which allows to rotate the network in the Net View area and to visualize it from the best view point.

In the Control Panel are also placed all the controls for the construction of a musical network, which can be created by the user by choosing the number of nodes, links per node and the percentage of randomness in the links' distribution. A network (constructed with SWAP or loaded from a MIDI file) may be saved as a text file which represents the network as a matrix. Each entry of this matrix has two indices related to the nodes of the network, and its numerical value represents the connections in the structure. So if we have 0 between two nodes there is no link between them, on the contrary if the value is 1, we have an edge that links the two nodes. This representation allows the user to define networks and to place each link in the structure finding the more appropriate configuration. The second functionality of SWAP is called Synthesis, and allows to create scales and melodies from the networks created or from those imported from MIDI files. This function will be better described in another section of this paper.

4 Graph-Based Analysis

The first step of our work consisted of thorough analysis of the topology of networks derived from musical compositions. The parameters analysed are the ones studies by Watts and Strogatz in their paper, clustering coefficient and path length. We extracted these data from the networks and then compared all these



C	BWV532	BWV564	BWV543	BWV526	BWV570	BWV572
Regular	0.69	0.7	0.7	0.69	0.6	0.66
Network	**0.61**	**0.63**	**0.68**	**0.62**	**0.5**	**0.52**
Random	0.16	0.21	0.21	0.32	0.005	0.16
L	BWV532	BWV564	BWV543	BWV526	BWV570	BWV572
Regular	3.24	2.8	2.8	2.17	5.5	3.5
Network	**2.47**	**2.33**	**1.9**	**1.88**	**3.24**	**2.58**
Random	1.87	1.8	1.79	1.7	2.42	1.97

Fig. 2. Comparisons between C and L values of Bach's compositions with random and regular networks with the same number of nodes and edges

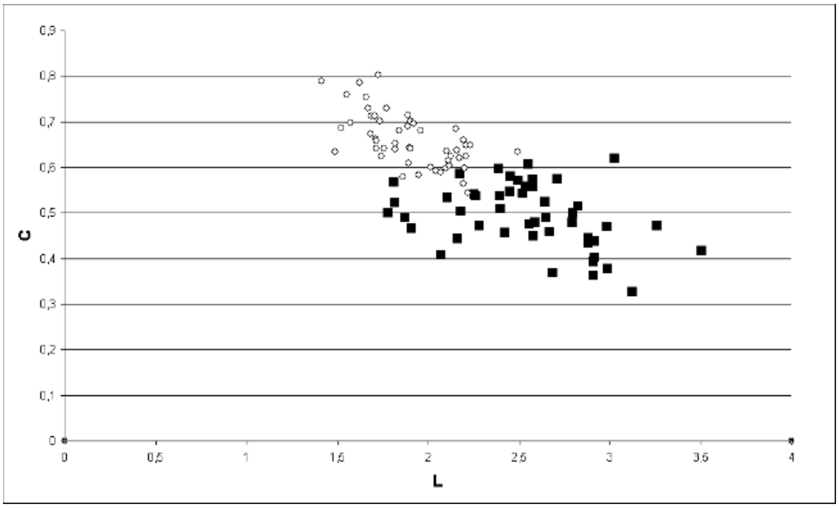


Fig. 3. L-C Space. Networks extracted from Bach's (squares) and Mozart's (circles) composition. Each symbol in the L-C space is characterized by values of clustering coefficient and average path length.

values with the values of completely ordered and random networks with the same number of nodes and edges. The results are very interesting, since the data of our networks are found to be intermediate between order and randomness. Fig. 2, shows some data extracted from Bach's compositions and some graphical representations of these comparisons.

The second step was to compare values from different authors trying to find interesting differences that can be underlined by means numerical and graphical data. We introduced a two-dimensional space with path-length on X axis and clustering coefficient on Y axis (Fig. 3), and found a certain tendency of different authors' compositions to occupy different areas of this space. There are of course some overlaps in these data. Although these data are only preliminary, but we also find them interesting.

Finally, we carried out an ANOVA analysis between groups of compositions of different authors. We compared C and L values from about 100 musical compositions classified for musical genre (Classical or Pop) and author (J. S. Bach, F. Battiato, The Beatles, G. F. Handel, W. A. Mozart and F. Zappa). The first analysis did not produce significant results, there was no significant difference between data from different genres, but the second one showed significant differences in the clustering coefficient of different composers:

$$F(5, 95) = 4.461, p < .001$$

this result may be a good starting point for more analytical studies and for the development of graph-based algorithms for author attribution.

5 Creation of Melodies

There are two basic algorithms for the production of melodies through SWAP. The first finds the minimum path between given start and an end nodes, generating a simple melody. This function, based on Floyd-Warshall algorithm, is basically a way of exploring network's characteristics through sound. From a musical view point, the scales derived from this technique are not very interesting, but we found this algorithm a very interesting to better understand the dynamics of the studied network.

The second is a simple random algorithm called Walking which, starting from a given node, randomly chooses the path to cover by randomly selecting subsequent links. The user may define the start node, the sequence's length, the speed (expressed in beat per minute) and the resolution of the melody (1/4, 1/8, 1/16 etc.). The melodies obtained by this algorithm are much more pleasant than the previous ones. In this case, the melody does not follow a minimum path but is "free to walk" in the structure. This method may show some similarities with the Markovian Chains, but walking is a simpler algorithm. Markovian Chain is a stockhastic model in which the transition probability which determines the passage to another state of the system depends only on the previous state. Walking is a model in which the selection of the paths is based on the distribution of links in the structure. Thus, if we have as fist step a node connected to three other nodes, the three respective links have the same possibility of being choosen. The only constraint is the connection between vertices, and the paths are possible only where a link is placed (there is no possibility of going, for instance, from a generic node A to another node B if there is no link between them).

Of course, the resulting melody is influenced by the distribution of the edges in the network, so that if we have interesting intervals in the network the scales will be more music-like. For instance, also the "imported" networks can be re-used in order to create melodies. This is an interesting technique because allows to use a structure whose melodic intervals (i.e., the connections between nodes) are of course non-dissonant, so the resulting scale should be melodic.

The interesting thing is that the melody produced by means of this technique are completely different by the original one. As we discussed above, network is a kind of representation which can be thought of as a map, so a virtually infinite number of musics can be created by the same network, just like there are infinite ways of walking in the streets of a city. Since our algorithm is completely random the musics generated are self-organized, so we do not even influence musical creation.This result suggests that a "good" network is able to produce nice melodies just by means of a random algorithm. In the next section we will introduce the use of genetic algorithms for the generation of networks with determined characteristics in order to produce interesting melodies.

6 Genetic Algorithms

As we discussed above, we created a representation in which musical compositions are characterized by points in a L-C two-dimensional space. Since we noted the tendency of compositions of different authors to occupy different areas in the space, we decided to implement a system for the creation of "author-like" networks, based on genetic algorithms. Roughly speaking, we can inscribe a region of the space in a circle of radius r, and decide that the compositions contained in that area are the ones we are looking for.

Then we use genetic algorithms with the purpose of seeking graphs located near the center of the circle. So to do this we defined a fitness function for minimizing of the distance from the center of our interested area. For instance, let P be a point in the L-C plane, with distance d from the center T of our area. We may define the fitness function as follows:

$$f(d) = 1 - d^{\frac{log(1-t)}{log(r)}} \tag{3}$$

where t is the threshold parameter. Fig. 4 shows the basic features of our fitness function. After defining an interesting region, for example by observing the distribution of compositions of a certain author in the L-C plane, it is possible to grow populations of networks situated near the center of the circle. In order to create a population of networks we define the number of individuals of the population and the number of nodes and links of networks. As we said we created a fitness function for minimizing the distance from a target point called T

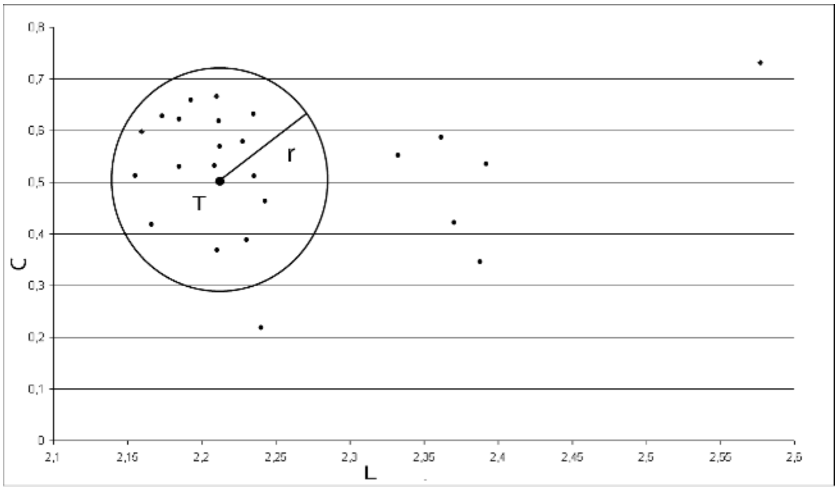


Fig. 4. Graphical representation of fitness function target

in the L-C plane, of course the target is characterized by values of average path length (X axis) and clustering coefficient (Y axis). Thus we define the following parameters: Target Point (i.e., the C and L values of the networks), Random Rate (the percentage of new fellows), Injection Rate (the number of epochs for each random injection of individuals), Mutation Rate (the probability to have a mutation of a gene), Elite (the percentage of individuals that will be taken for the next population), Crossover Rate (the number of individuals that will be genrated by crossover), Radius (the radius of the target area) and the Number of Epochs. Fig. 5 show results of some experiments on a population of 300 fellows for 1000 epochs. We defined a random rate of 5% , an elite of 2%, a crossover rate of 70% and a mutation rate of 0.1%. The initial population has an average fitness just above 0.3 and the best individuals are very close to the average, after 1000 epochs the best networks have more than 0.8 of fitness and the mean is over 0.75. As shown in figure, the introduction of new individuals generates a decrease of the mean, in fact the mean's values are continuosly oscillating. In this case we fixed the injection rate at 10, so every 10 epochs a 5% of new fellows are inserted in the population.

7 Conclusions and Further Developments

This work shows a new approach on musical representation and analysis based on small-world networks. Musical pieces have been represented as graphs and studied by using the tools of science of networks. The results obtained with these experiments are interesting since they underline similarities between the inner structure of some compositions and the graphs studied by Watts and Strogatz and by Barabási. Of course, this is the first step of a more complex study ori-

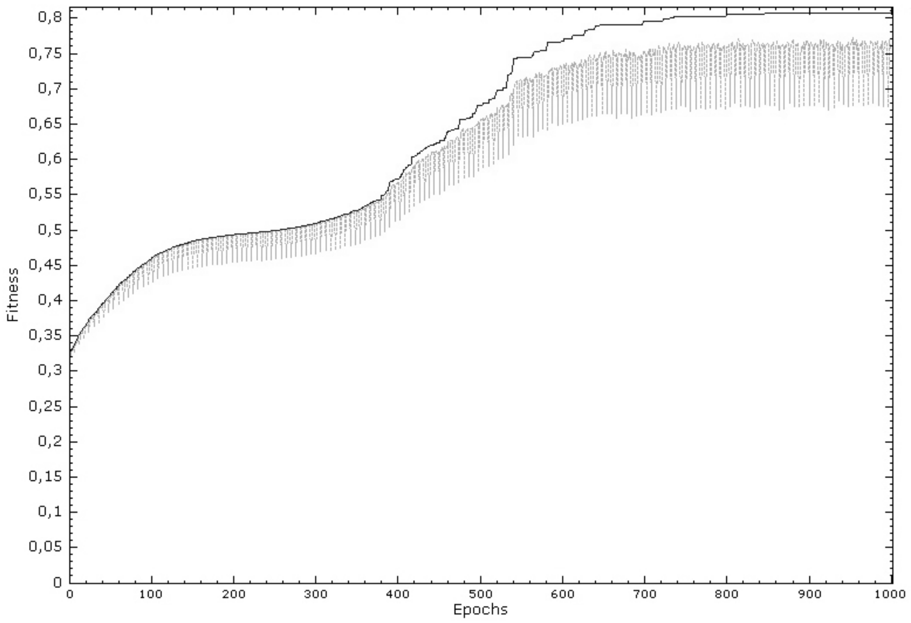


Fig. 5. Results of experimentations with genetic algorithms. The black line represents the fitness values of the best fellows and the grey line represent the mean.

ented to generative and evolutionary music, computer assisted composition and computer-based musicology.

Our future work will be basically focused on the deepening of our statistical analysis, and on other ways of graph-like representations (for example, using hubs and scale-free networks). Other experimentations will be oriented to find useful criteria for defining pleasant music and to use them to build appropriate fitness functions for the construction of musical networks. We are trying to use Zipf's law [21, 13, 22, 23] for the evaluation of music. Moreover, we are going to define more complex and sophisticated algorithms for musical creation and sound synthesis, both with and without human interaction.

References

1. Watts, D. J., Strogatz, S. H.: "Collective Dynamics of Small-World Networks". Nature 393 pp 440-42 (1998).
2. Newman M. E. J.: "Clustering and Preferential Attachment in Networks". Phisical Review E, p. 25102 64 (2001).
3. Newman M. E. J.: "Models of the Small World". A Review , J. Stat. Phys. 101, 819-841 (2000).
4. Newman M. E. J., Watts D. J.: "Scaling and Percolation in the Small-World Network Model". Phys. Rev. E 60 7332 (1999).

5. Albert R., Barabàsi A. L.: “Statistical Mechanics of Complex Networks”. Reviews of Modern Physics, 74 (1) pp. 47-97 (2002).
6. Albert R., Jeong H., Barabàsi A. L.: “Diameter of the World Wide Web”. Nature, vol. 401 pp. 130-131 (1999).
7. Albert R., Jeong H., Barabàsi A.L.: “Error and attack tolerance of complex networks”. Nature, 406, 378 -382 (2000).
8. Barabàsi A. L., Albert R.: “Emergence of scaling in random networks”. Science, vol. 286 pp. 509-512 (1999).
9. Jeong H., Neda Z., Barabàsi A.-L.: “Measuring Preferential Attachment in Evolving Networks”. Europhys. Lett. 567572 (2003).
10. Latora V., Marchiori M.: “Efficient Behavior of Small-World Networks”. The American Physical Society, volume 87 issue 19 (2001).
11. Latora V., Marchiori M.: “Economic Behavior in Weighted Networks”. Eur. Phys. J. B 32, pp. 249-263(2003).
12. Marchiori M., Latora V.: “Harmony in the Small-World”. Physica A 285, pp. 539-546 (2000).
13. Manaris B, Romero J.: Machado P., Krehbiel D., Hirzel T., Pharr W. And Bavis R. B.: “Zipfs Law, Music Classification and Aesthetics”. Computer Music Journal, 29:1, pp 55-69, MIT Press (2005).
14. Weinberg, G.: “Interconnected Musical Networks: Toward a Theoretical Framework”. Computer Music Journal, 29:2, MIT Press, pp 23-39, Summer 2005.
15. Cage, J.: Silence, Middletown, CT: Wesleyan University Press (1961).
16. Campolongo G., Vena S.: “Complessitá e Musica. Analisi e Generazione di Melodie con Reti Small-World”. Proceedings of the Conference Mathematics, Art and Cultural Industry May 19-21 2005. In Bertachini P. A., Bilotta E., Francaviglia M., Pantano P. (Eds.), Cetraro (2005).
17. Campolongo G., Vena S.: “Analysing and Creating Music through Small-World Networks”. Proceedings of the V International Conference Understanding and Creating Music, Seconda Universitá di Napoli, November 27-30 Caserta (2005).
18. Campolongo G., Vena S.: “Applications of Small-World Networks to Music”. Accepted for Generative Art Conference, December 15-17 2005 Milan (2005).
19. Bilotta E., Pantano P., Talarico V.: “Synthetic Harmonies: An Approach to Musical Semiosis By Means of Cellular Automata”. In Bedau M.A., J.S. McCaskill, N.H. Packard, S. Rasmussen (Eds.) Artificial Life VII, August 2000. Cambridge, Massachssets. The MIT Press (2000).
20. Bilotta E., Pantano P. Talarico V.: “Music Generation Through Cellular Automata: How to Give Life to Strange Creatures”. Proceedings of Generative Art GA 2000, Milan, Italy (2000).
21. Manaris B., Vaughan D.: Wagner C., Romero J., Davis R. B.: “Evolutionary Music and the Zipf-Mandelbrot Law: Developing Fitness Functions for Pleasant Music”. Proceedings of EvoMUSART2003 1st European Workshop on Evolutionary Music and Art. Berlin: Springer-Verlag (2003).
22. Mandelbrot, B.: “Information Theory and Psycholinguistics: a Theory of Word Frequencies”. Readings in Mathematical Social Science. Lazarfeld P., Henry N. (Eds.), Chicago: Science Research Associates, 350-368 (1966) Cambridge, MA .
23. Zipf G. K.: Human Behavior and the Principle of Least Effort. Addison-Wesley, Cambridge, MA, (1949).

Continuous-Time Recurrent Neural Networks for Generative and Interactive Musical Performance

Oliver Bown[1] and Sebastian Lexer[2]

[1] Centre for Cognition, Computation and Culture
[2] Department of Music,
Goldsmiths College, University of London,
New Cross, SE14 6NW, UK
o.bown@gold.ac.uk, s.lexer@gold.ac.uk

Abstract. This paper describes an ongoing exploration into the use of Continuous-Time Recurrent Neural Networks (CTRNNs) as generative and interactive performance tools, and using Genetic Algorithms (GAs) to evolve specific CTRNN behaviours. We propose that even randomly generated CTRNNs can be used in musically interesting ways, and that evolution can be employed to produce networks which exhibit properties that are suitable for use in interactive improvisation by computer musicians. We argue that the development of musical contexts for the CTRNN is best performed by the computer musician user rather than the programmer, and suggest ways in which strategies for the evolution of CTRNN behaviour may be developed further for this context.

1 Introduction

At the junction between computer music and artificial intelligence lies the goal of developing generative or interactive software agents which exhibit musicality. The appearance of musicality is determined either by a listening, watching audience or an interacting performing musician. Attempts in this domain have taken on a wide variety of forms due, on the one hand, to the wide variety of methods in artificial intelligence, and broadened further by the myriad possible interpretations of these techniques in a musical domain, and in addition by the myriad musical styles and substrates in which such an interpretation takes place [1, 2].

The approach presented here is inspired first and foremost by a search for general-purpose behavioural entities that could be adopted by computer musicians in a flexible manner. Our approach is influenced by, and indeed made possible by, the availability and popularity of modular extensible computer music platforms such as Max/MSP [3], PD [4] and SuperCollider [5]. Practising musicians who work with these tools often build up personalised repertoires of software patches and commonly adapt publicly available third-party objects to their own performance needs. This feeds a powerful form of social creative search in which something designed with a given purpose in mind may be reappropriated indefinitely. Thus rather than thinking in terms of stand-alone intelligent musical systems, we conceive of a generic behavioural tool that can be developed in different directions by practising musicians.

This paper proposes the Continuous-Time Recurrent Neural Network (CTRNN) as one such tool. The remainder of this section gives a background and technical description of the CTRNN and discusses aspects of its behaviour that are relevant to musical improvisation. Section 2 discusses the implementation of the CTRNN in Max/MSP and initial performance uses of the CTRNN, and expands upon our methodology. Sections 3 and 4 discuss methods for evolving CTRNNs and for designing performance contexts for them.

1.1 Background

Non-symbolic artificial intelligence (AI) emphasises low-level behaviours at the heart of all species' strategies for survival, as understood in terms of Darwin's theory of evolution [6]. Early work in cybernetics by Grey-Walter [7] established an experimental context in which wheeled robots, containing sensors and motors connected by simple analogue circuits, could be designed to produce observably *lifelike* behaviour. Since the development of Genetic Algorithms (GAs) and increasingly smaller and faster computer processors, it has become possible to evolve compact algorithms that allow a physical wheeled robot to satisfactorily perform more precisely defined cognitive tasks.

The notion of *minimal cognition* [8, 9] has helped home in on the meaning of the term *lifelike*. In recent years a great effort has been made to understand how extremely simple biologically-inspired algorithms could learn tasks such as object recognition, selective attention and simple memory, using CTRNNs *embodied* in simulated agents and *situated* in simple physical environments.

1.2 Technical Description of the CTRNN

CTRNNs are a kind of artificial neural network: an interconnected network of simulated neurons modelled on a computer. In the case of CTRNNs neurons are typically of a type known as the *leaky integrator*. This is a greatly simplified model of a real neuron, with a continually updating internal state determined by a differential equation,

$$\tau_i(dy_i/dt) = -y_i + \sum W_{ij}\sigma(g_j(y_j - b_j)) + I_i \tag{1}$$

where τ_i is the time constant, g_i is the gain and b_i is the bias for neuron i, I_i is any external input for neuron i and W_{ij} is the weight of the connection between neuron i and neuron j. σ is a non-linear transfer function which in our case is *tanh*.

CTRNNs allow recurrency, meaning that network connections can exist in any direction, including potential connections from any node to itself. The combination of recurrency and internal state makes for a system which can produce complex internal patterns of activity and which has a memory-like response to its environment [8]. Each node has three parameters – bias, gain and time constant – associated with its behaviour, and each connection between nodes has

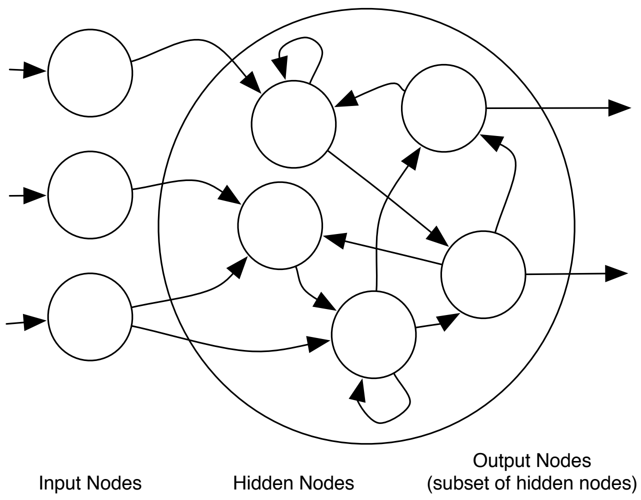


Fig. 1. CTRNN architecture

one parameter – a weight. Due to the complex relation between network parameters and network behaviour, along with the non-specificity of solutions to tasks that they are typically used for, a common method for arriving at a CTRNN which performs a certain task is to use a Genetic Algorithm (GA).

CTRNNs are known to be theoretically capable of replicating any dynamical system, and it has been shown that very small CTRNNs are capable of arbitrarily complex dynamics [10].

1.3 Categorisation of Behaviour from CTRNN

Our interest in the CTRNN as a musical unit stems from the potentially unbounded range of temporal dynamics of which it is capable. It is a sub-symbolic system, meaning that it is unlikely to have immediate application in the domain of discretised musical events traditional to computer music and fundamentally implicit in human musical behaviour. Although an appropriate use of CTRNNs would therefore be in the signal domain, we focus on simple rhythmic behaviours at the control rate, and we refer to this domain as gestural. The CTRNN's interaction with the world consists of vectors of real valued inputs and outputs, updating continually and frequently, in our case on the order of 10 milliseconds. A representative example of the CTRNN in a musical context, which illustrates our intended use of the system, places it with a series of features extracted from an audio stream as input, and a set of synthesis parameters as output. In this case the CTRNN is conceived of as a direct interactive participant in a musical performance.

Beer [10] provides an extensive mathematical analysis of the behaviour of small CTRNNs, the presentation of which is beyond the scope of this paper. A key notion in such an analysis is that nodes with suitably strong self-connections can *saturate*, meaning that their own feedback dominates their input and locks them in a certain state. In this way nodes can act as internal switches which influence the behaviour of the rest of the network. Such nodes may flip states. More broadly, due to its internal state and recurrency, the state of a CTRNN at time t is determined not only by the present input, but the history of inputs up until time t, and the starting state of the network.

In the case of a static input (*i.e.*, whose values are not changing), CTRNNs can be described in terms of their internal dynamics in the same way as any closed dynamical system. The network will either find a static resting state, move periodically, or move quasiperiodically or chaotically [11].

In the case of changing inputs, we consider three distinct categories of CTRNN behaviour *as perceived*. These categories are specific to our musical purpose and are not derived from a formal approach: they attempt to capture a musician's perception of CTRNN behaviour rather than CTRNNs themselves.

In the first category, each input state leads to one output pattern, which may be either static or cyclical. As the input state moves from A to B and back again, the CTRNN moves between associated patterns, returning to the same pattern for each input.

The second category is identical to the first except that after changing input state from A to B, a transitory period of indeterminate length precedes the output's arrival at its resting pattern. The length and form of these transitionary sections vary depending on the particular trajectory through input states, but always end up at the same pattern for any given input state.

In the third category, there may be more than one output pattern for each input state, and which one is arrived at will be determined by the trajectory through input states. Thus moving from input state A to input state B leads to a different output pattern than if one were to move via input state C. In other words, the system has multiple attractors for the same resting input, and is dependent on the history of the input leading up to its resting state.

2 Musical Uses

The use of neural networks in studies of music cognition and in composition already has a long and rich history. Commonly cited benefits of a neural network approach to musical problems are *generalisation*, the possibility of extrapolating general features from a corpus of examples, and *graceful degradation*, the robust nature of the network's response to unexpected inputs, as compared to rule-based systems. [12] and [13] contain an exemplary cross-section of such work. Mozer [14], for example, uses trained recurrent neural networks to generate musical melodies with note-by-note prediction. He identifies and addresses problems of securing musical structure over longer time scales than are naturally dealt with by the network, and thus improves the quality of melodies generated in this way. In this and other work in music and AI the final goal is often a machine that makes novel competent music within a given context without the aid of a musician. According to the valuable 'Frankenstein' analogy provided by Todd and Werner in the same volume [1], the present approach differs from this body of work by beginning with the problem of the 'monster's' acceptance in society.

2.1 The CTRNN as Max/MSP Object

The first author has implemented the CTRNN in C as a Max/MSP external object. The object can read and write networks to and from plain text files, and can also randomly generate networks according to a set of user-specified parameters. It can also save and recall network states.

Inputs are sent to the CTRNN as a list of floats to the object's left inlet. Each input list triggers one update of the network, causing a list of output values to be sent from the object's left outlet. Inputs should therefore be sent at equal time intervals. For behaviour suitable for human interaction, a rate of the order of 10 milliseconds is appropriate. This can easily be achieved in a robust manner in Max/MSP.

We describe below how networks with specific behaviour have been generated using a GA in a simple command-line program so that the Max/MSP object can then read them. In other cases, users can randomly generate networks by selecting a number of input nodes, a number of outputs, and a number of internal (hidden) nodes, as well as setting maximum and minimum values for a number of parameters that will then be generated at random. These ranges include values for the time constants, biases and gains for both hidden nodes and input nodes (separate ranges are allowed for each type of node), for weights of connections from input nodes to hidden nodes, and finally for weights of connections from hidden nodes to hidden nodes. There is also a *density* parameter which determines the proportion of connections that are non-zero. Randomly generated networks obviously do not have precisely pre-specified behaviour, but they still obey tendencies. For example it is easy to vary the number of hidden nodes and the density of networks and observe an increased activity as both of these values are increased.

2.2 Development of a Methodology

Initial motivation for using the CTRNN came from a desire to generate metrically free rhythmic patterns, inspired by the work of Karl Sims in evolving locomotive behaviours [15]. Rhythmic patterns were achieved by placing a threshold on one of the CTRNN outputs in order to trigger drum events. This idea was the subject of the first author's MSc dissertation, and was conducted entirely in a non-realtime context.

The second author, working with freely improvised electroacoustic music, has started working with a trial version of the CTRNN Max/MSP object to control a spectral filter through which he plays the piano. This reframes the original motivation behind using the CTRNN according to a distinction in electroacoustic theory between the systems autonomy of the instrument and the control of the performer. According to this the most interesting period in the case of the piano is its decay period, which lies beyond the control of the player: the sound is on its own after the hammer strikes the string. The pianist John Tilbury describes this as the *contingency* of the piano sound [16]. The present form of the CTRNN, even randomly generated, offers an opportunity to implement similar contingent relationships between performer and electroacoustic process, generating micro structures that control electroacoustic processes in turn dependent on the performer's activity. Given the CTRNN as a Max/MSP object the user would be free to develop his or her own approach to the problem of how to map inputs and outputs, and take on the responsibility of making the CTRNN sound good in his or her own musical context through an iterated approach of performance and adjustment. The challenge of refining parameter mappings is already familiar to any musician developing interactive software.

Introducing evolution to the project suggests the possibility of developing a system that has certain behavioural facets that would drive an engaging improvisation in very basic ways: for example responding to an input pattern but not in the same way every time; behaviour potentially changing over time; settling into patterns but producing variations on themes. Put more loosely, such a system should have idiosyncrasies and tendencies that the user could put to effective use to drive an improvisation. However, writing fitness functions that result in these desires being met is not straightforward. In the following section we describe initial attempts to do this.

3 Evolving Musical CTRNNs

Our long term aim is to develop methods for allowing musician users to be able to define their own behavioural targets for CTRNNs. However, since the use of GAs is not straightforward, some thought needs to be given to how this is facilitated, and what kind of behaviours can be defined. This paper deals with a simple initial experiment to evolve CTRNNs that exhibit the third category of behaviour in section 1.3, in the hope that the resulting networks exhibit behaviour that is inherently appealing to musicians.

This only goes half way to the claim of evolving musical behaviour since there are many musical considerations left over. In the previous section, we argued that the CTRNN should be placed in a musical context by the musician user. In the language of behavioural robotics, this means it is up to the musician to embody and situate the CTRNN. Embodiment, in this context, refers to the CTRNNs set of input and output systems in Max/MSP, such as audio analysis tools at the input, and synthesiser parameters at the output. Situatedness, in this context, refers to the musical environment in which the CTRNN will be used, which includes the playing styles and knowledge of the performers who will be playing with the CTRNN. Note that whilst, ideally, behavioural systems are evolved in embodied, situated contexts, the complications that come with achieving this in a musical context are *side-stepped* by trying to evolve general-purpose musical behaviours. Such issues are addressed in the concluding remarks.

CTRNNs are assumed to be fully-connected in the hidden layer, and with a full set of connections from the input layer to the hidden layer. Non-fully connected CTRNNs are thus expressed as having some zero weights. The number of parameters needed to describe a fully-connected CTRNN with n hidden nodes and m input nodes is $n^2 + 3n + nm + 3m$ (since there are n^2 connections between the hidden nodes, and nm connections from the input nodes to the hidden nodes, each with a single weight, as well as bias, gain and time constant parameters for each of the hidden and input nodes). Genotypes for the CTRNNs were expressed as vectors of real numbers in the range $\{0,1\}$, with a mapping from genotype to phenotype that transformed this range to prespecified ranges for each of the parameters. All mappings were linear except for time constants, which were mapped exponentially.

A rank-based GA was used with standard crossover and mutation. At each generation, pairs of individuals from the fittest third of the population were selected at random for breeding, and their offspring were used to replace individuals from the weakest third of the population. Mutation involved adding a random vector of average length 0.01, drawn from a Gaussian distribution, to the individual genotype string. Genotype values that fell outside of the range $\{0,1\}$ were *bounced* back in with an elasticity of 0.1. Multipoint crossover was used, with a probability of 0.1 that the source genotype would be swapped at each point along the genotype.

The fitness function for the CTRNN was as follows: a number x of input patterns were generated using random walks starting from the origin (the same set of input patterns were then used for all of the trials). For each input pattern the CTRNN was reset to zero, and run on the input values for t time steps, then on a fixed input of zero for another t time steps, and then for another $t/10$ time steps, still with the zero input, the output sequence for this final period being stored. The x stored output sequences were then compared with each other in order to establish their similarity. For each pair of outputs, absolute differences between corresponding pairs of values in the output sequences were taken and summed to produce a dissimilarity score. This was repeated with different time offsets, and the lowest possible dissimilarity score was taken (*i.e.*, the score for the case in which the pairs were most similar). The average of the dissimilarity scores for each pair of outputs was taken as the fitness of the CTRNN. This meant that out of phase, but otherwise identical, periodic outputs would receive a score of zero. Two versions of the evolutionary process were attempted. In the first a gradual increase in the number x of input sequences over the course of the GA was used in order to smooth the difficulty of the task. Early on in the GA CTRNNs had tasks involving 3 input sequences, and over time this was increased to 20 input sequences. In the second version, 20 input sequences were used right from the beginning. GAs were run for 2000 generations. t was fixed at 500.

A number of evolutionary runs were made with different values for the number of hidden nodes from 3 to 10. In each case networks were give two inputs and three outputs. This was so that the network could be later interacted with using a two-dimensional controller in Max/MSP (hence the two inputs) and could be visualised in a three-dimensional parameter-space plot (hence the three outputs).

The resulting CTRNNs were tested for the generality of their behaviour by being compared with 200 random CTRNNs of the same number of nodes on an identical task as the fitness function, but with a new set of random input patterns which were different from the ones used in the fitness function. In other words, the CTRNNs were tested to see how far their history-dependent behaviour extended. Figure 2 shows the results of such a test averaged over 10 trials. A few random nodes score higher than the evolved nodes. It was noted that random nodes with a higher density of connections scored higher, exhibiting more erratic behaviour in general. However, despite the fact that the performance of the evolved CTRNNs is quite variable, the graph indicates that the evolved behaviour has some generality.

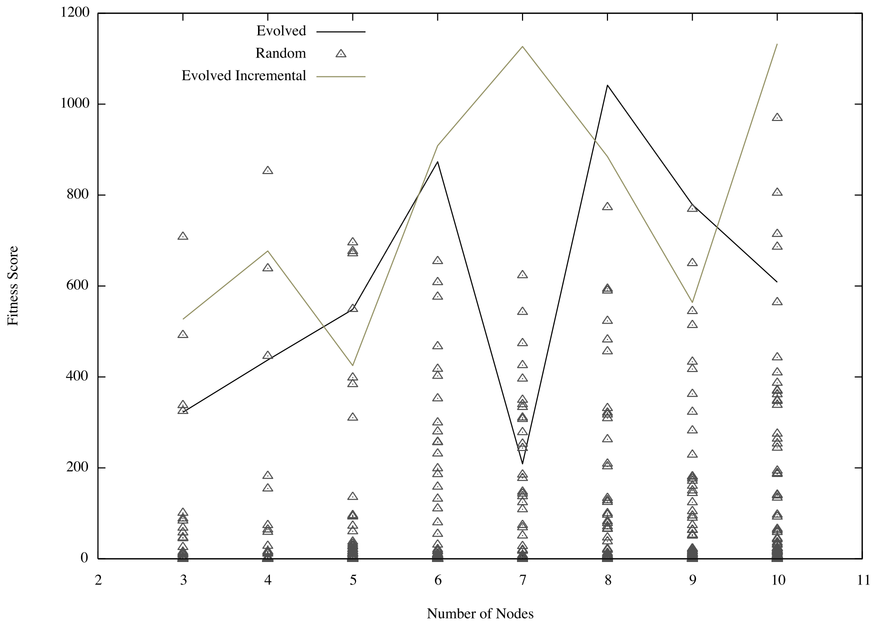


Fig. 2. Generality of fitness of evolved networks (lines) versus random networks (points) for a range of node numbers. The lighter line shows networks evolved using an incremental fitness function.

Informal interactive testing of the evolved and random networks showed a remarkable range of behaviours, and the most immediate implication of this is that a more thorough categorisation of CTRNN behaviour from a musical point of view is in order. Evolved networks tended to be doing something more *interesting* and often the nature of their evolved behaviour was immediately apparent: by repeating the same input patterns one could easily observe the network falling into distinct cyclic attractors. The incrementally evolved networks seemed to find less satisfying solutions which often involved oscillating at a rate near to the time step of the CTRNN update rules; a behaviour that it would probably be beneficial to punish. Larger networks were generally more interesting, but the best networks from this trial, as judged by the first author, were the 5 and 10 node evolved networks, suggesting that larger networks are not necessarily more interesting.

Similar informal tests with a wider set of CTRNNs revealed the potential for compositionally useful behaviour in almost all networks, both random and evolved: regardless of interactive potential or sustained varied dynamics, many of the oscillations made by CTRNNs at cyclic attractors were rhythmically and dynamically pleasing when connected up to various synthesis or filter parameters.

We have used the word 'interesting' without hope of qualifying it with *why* things are interesting at this stage. More formal tests should be set up to determine which kind of network behaviours are of interest to a range of performing musicians. Since our focus is on interactive behaviour in an improvisational context our main concern is whether the CTRNNs can sustain interest through interaction. Whilst our small evolved CTRNNs had a relatively predictable behaviour (although they produced more sustained interest than random networks of a similar size), more complex CTRNNs (with large numbers of densely connected nodes) produced complex output patterns that were intriguing to listen to but appeared to have very little to do with what was being played to them, thus failing to be interactive. Truly interesting interactive behaviour comes with the sense that the CTRNN is responding to its input *at the same time as* performing autonomously. This has been hard to pinpoint in the present study but should be made a central concern in future tests.

We conclude that it is relatively easy to define targets for CTRNN behaviour and approach those targets through evolution, but that further investigation is needed into *how* to specify target behaviour that is musically useful. Now that a framework is in place in which CTRNNs can be evolved and tested, both in simulation and through direct interaction in musical contexts, it will be possible to extensively explore a variety of approaches to designing fitness functions and applying network behaviours to musical goals. Plans for future work in these areas are discussed in the following section.

4 Future Work

4.1 Finding Target Behaviours

With the standalone GA application in place, exploring behaviours evolved for different tasks becomes a simple matter of writing new fitness functions. The fitness function used in the experiment above aims to produce a behaviour that exhibits some minimal aspect of musicality. A range of similarly general purpose fitness functions could be added to this to produce a repertoire of behavioural units, and to this extent it would be desirable to test a variety of simple fitness

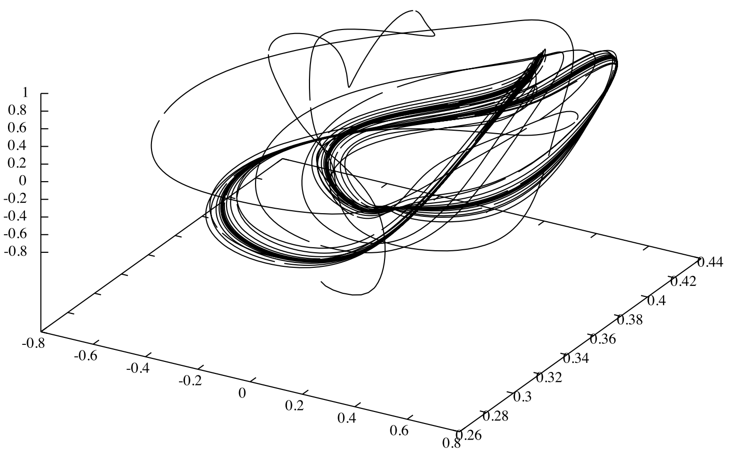


Fig. 3. Example of an output trajectory for an evolved CTRNN

functions in an iterative manner with improvising musicians. Alternative functions could focus on levels of predictability, rhythmic features of the CTRNN behaviour, call and answer dynamics, or producing specific patterns under specific circumstances.

A natural development would be to get CTRNNs to imitate human performances, beginning with recorded training data which the CTRNN is expected to imitate. Referring back to our representative example in section 1.3, a performer could take over the part of the CTRNN, controlling the filter, and all input and output data could then be recorded. However, this approach implies difficulties: it would not be sufficient just to feed the CTRNN with the input sequences and rate it according to how close its output comes to the target output sequences. At the very least the sequences would need to be divided into a set of discrete trials; this is necessary in order to convince the network to pay any attention to its inputs. To this end it will be necessary to explore how data sets can be recorded and divided into significant events either manually or automatically.

4.2 Developing Mappings from CTRNN Outputs

When using the CTRNN in Max/MSP, the user is able to observe three dimensions of output states from a CTRNN in a window generated in Jitter, the video editing extension to Max/MSP. By observing output states during practice we propose developing this interface so that a musician could draw colour-coded regions into the 3-D space that he or she wishes to correlate to specific parameter settings of an instrument. A feedforward neural network could then be trained to implement the desired mapping. Through an iterative process of practice and adjustment a more carefully crafted combination of behaviour and desired sound could be developed, bringing together a CTRNN behaviour with a specific repertoire of output states. Successfully implementing this addition may reinforce the notion that it is sufficient to provide the musician with a set of CTRNNs with general-purpose behaviours. The musician is then able to chose from a set of behaviours, and iteratively design a mapping from behaviour to audible musical output. Similar processes could be applicable to the input of the network.

4.3 Evoking a Coevolutionary Context

Better still would be to create a context in which the behaviour of the CTRNN can be modified in real time, so that the CTRNN, its embodiment (the Max/MSP input-output context), and its situatedness (the musical context that it will be used in) could collectively converge, rather than the latter two converging around the former. Thus experimentation with real-time interactive CTRNNs begs the notion of a coevolution between user and unit that would lead to a powerful interactive system at the moment of performance, but would also imply a gradual adaptive development of all aspects of the system during preparation. CTRNNs can also be modified in various ways to introduce *ontogenic adaptation* (as opposed to evolution) into the preparation process. This paper stops short of suggestions for how these developments could be achieved, but research into this problem could take various immediate directions likely to throw up fruitful results.

5 Summary

We have introduced the CTRNN as a performative and/or compositional tool for musicians using modular extensible computer music platforms such as Max/MSP. We have described how networks can be randomly generated or evolved to produce particular behavioural properties, and demonstrate very simple examples in which evolved CTRNNs exhibit behaviours that are of interest to improvising musicians.

We have discussed future work in this area, including gathering training data to be used for the evolution of more specific CTRNN behaviours and developing mappings from CTRNNs to performance parameters using a trained feedforward network. We suggested that the CTRNN should be adapted by musicians according to their own performance contexts and their own interpretation of its behaviour, and that it should inform their own actions during performance as well as during the development of their performance contexts.

The notion of a coevolution or adaptive codevelopment between CTRNN behaviour and user is provoked by the present work. We suggest that this problem could be made into a fruitful topic of research.

Acknowledgments

Oliver Bown's research is supported by a bursary from the Department of Computing, Goldsmiths College. We would like to thank Geraint Wiggins for extensive feedback and Mike Riley and the Goldsmiths Department of Visual Arts' Digital Media Lab for the use of their computers as Apple XGrid agents.

References

1. P. M. Todd and G. Werner. Frankensteinian approaches to evolutionary music composition. In Niall Griffith and Peter M. Todd, editors, *Musical Networks: Parallel Distributed Perception and Performance*, pages 313–339. MIT Press/Bradford Books, Cambridge, MA, 1999.
2. E. Miranda. *Composing Music with Computers.* Focal Press, 2001.
3. http://www.cycling74.com.
4. http://www.puredata.info.
5. http://www.audiosynth.com.
6. C. Darwin. *On the origin of species by means of natural selection, or The preservation of favoured races in the struggle for life.* D. Appleton and company, 1860.
7. W. Grey Walter. An imitation of life. *Scientific American*, 182(4):42–54, 1950.
8. A.C. Slocum, D.C. Downey, and R.D. Beer. Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention. In J. Meyer, A. Berthoz, D. Floreano, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 430–439. MIT Press, 2000.
9. R.D. Beer. The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11(4):209–243, 2003.
10. R. D. Beer. On the dynamics of small continuous recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.
11. D. Kaplan and L. Glass. *Understanding Nonlinear Dynamics.* Springer-Verlag, 1995.
12. P. M. Todd and D. Gareth Loy. *Music and Connectionism.* MIT Press, 1991.
13. N. Griffith and P. M. Todd, editors. *Musical Networks.* MIT Press, 1999.
14. M. C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.
15. K. Sims. Evolving 3d morphology and behaviour by competition. In *Artificial Life IV Proceedings.* MIT Press, 1994.
16. J. Tilbury. Feldman and the piano: the art of touch and celebration of contingency. In *Second Biennial International Conference On Twentieth-Century Music, Goldsmiths College, University of London*, 2001.

Synthesising Timbres and Timbre-Changes from Adjectives/Adverbs

Alex Gounaropoulos and Colin Johnson

Computing Laboratory,
University of Kent,
Canterbury, Kent, CT2 7NF,
England
ag84@kent.ac.uk, cgj@kent.ac.uk

Abstract. Synthesising timbres and changes to timbres from natural language descriptions is an interesting challenge for computer music. This paper describes the current state of an ongoing project which takes a machine learning approach to this problem. We discuss the challenges that are presented by this, discuss various strategies for tackling this problem, and explain some experimental work. In particular our approach is focused on the creation of a system that uses an analysis-synthesis cycle to learn and then produce such timbre changes.

1 Introduction

The term *timbre* is used in various ways in music. One way is in describing gross categories of sounds: instrument types, the sound of certain combinations of instruments, different stops on a pipe organ, a discrete choice of sounds on a simple electronic keyboard, and so on.

A second aspect of timbre is the distinctive sound qualities and changes in those qualities that can be produced within one of those gross categories. To a skilled player of an acoustic instrument, such timbral adjustments are part of day-to-day skill. A notated piece of music might contain instructions concerning such timbres, either in absolute terms ('harshly', 'sweetly') or comparative terms ('becoming reedier'), and musicians use such terms between each other to communicate about sound ('Can you sound a little more upbeat/exciting/relaxed').

From here onwards we will denote these two concepts respectively by the terms *gross timbre* and *adjectival timbre.*

The player of a typical electronic (synthesis-based) instrument does not have access to many of these timbral subtleties. Occasionally this is because the synthesis algorithms are incapable of producing the kinds of changes required. However in many cases this lack of capability is not to do with the capacity of the synthesis algorithm—afterall, a typical synthesis algorithm is capable of producing a much larger range of sound changes than a physically-constrained acoustic instrument—but to do with the *interface* between the musician and the instrument/program [1, 2]. In current systems, the know-how required in order to effect the timbral change suggested by an adjectival description of timbre-change is vast.

Providing tools for manipulating timbre is an underexplored problem in computer music. In this paper we will discuss ongoing work on a project that aims to combine machine learning methods for searching synthesis parameter space and classifying timbre, together with analysis methods such as spectral analysis and principal component analysis. The long-term aim of this project is to produce systems that:

– Allow the synthesis of timbral changes to a sound from natural language descriptions of the desired change.
– Facilitate the automated discovery of transformations in synthesis parameter space that have meaningful timbral effects in sound space.
– Providing a framework whereby advances in the computer-based analysis of timbre can be used automatically to synthesise timbre and timbral changes.

2 Approaches to Timbre

2.1 Theory and Notation of Timbre

Compared to other aspects of music such as pitch and rhythm, timbre is not well understood. This is evidenced in a number of ways. For characteristics such as pitch and rhythm, there exist theories of how they work and produce perceptual effects; there are well-understood notations for them; and we understand how to synthesize them from fundamental components to get a particular effect.

By contrast, timbre lacks this repertoire of theory and notational support (as explored by Wishart [3]). Nonetheless there is a large repertoire of language associated with timbre and timbral changes. These timbral adjectives and metaphors provide a powerful means for musicians to communicate between themselves about timbre; but by contrast to the more formal notations for, say, pitch or rhythm, they do not provide a usable structure for inputting desired timbres or timbral changes into computer music systems [4, 1, 5, 6].

One approach would be to come up with a new language for timbre, which is more closely aligned with the way in which timbre is generated in electronic instruments. However this has many problems. For example timbre words convey information that has musical meaning, and we would like to create systems so that electronic and acoustic instruments can be played side-by-side and the players able to communicate using a common vocabulary. For these reasons we focus on how we can use traditional timbre words in a computer music setting.

2.2 Timbre as Gross Categorisation

At the beginning of this paper we introduced two notions of timbre: timbre as a gross categorisation of sounds, and timbre as the differences in sound qualities within those gross categories.

These two aspects of timbre are very different; most of the literature on timbre in computer music has focused on the gross categorisation, beginning with the early of Wessel [7].

An example of studies of gross timbre is the work of McAdams *et al.* [8]. In this work three dimensional timbre space was defined, the dimensions being attack time (time taken for volume of a note to reach maximum), the spectral centroid (the relative presence of high frequency versus low-frequency energy in the frequency spectrum), and spectral flux (a measure of how much the spectral changes over the duration of a tone). A number of instruments where then analysed by these three techniques and a graph of the results showed how each occupied a different part of the timbre space.

Such representations are useful when the aim is purely *analytic*, i.e. we want to understand existing sounds. However the aim of our work is oriented towards *synthesis*, and so we need to consider what representation is appropriate for being used 'backwards' to go from analysis to synthesis. Whilst a representation such as the three-dimensional model in [8] might yield acceptable results for categorising sounds, this representation is not adequate for synthesis of sound. We certainly could not work backwards from a three dimensional timbre representation and hope to synthesise the starting sound, since the representation is oversimplified and too much information has been lost.

Much of the recent work in the area of gross timbre has focused on the developing MPEG-7 standard. This work defines a framework for describing sounds in terms of spectral and temporal measurements of the sound, extracted through some analysis method. This work is interesting in that it identifies a number of features that are proven to be important for recognition and perception of timbre based on past research.

A large proportion of other research into timbre in computing has focused on automated recognition or categorisation of instruments. For example Kostek [9] describes a system that uses machine learning methods to classify which instrument is playing.

This has possible applications in databases of music for automated searching of stored sounds. The automated approach eliminates the need for a human to enter metadata identifying each sound, thus greatly simplifying the process of creating large sound databases. The common approach is to use neural networks to learn the sound of various instruments after being presented with various recordings of each. The key in this sort of work is to find common features between different recordings of a certain type of instrument, where the recordings may have different pitches, loudness, or playing style. Such features may be specifically symbolically represented, e.g. if the classification is performed using a decision tree method; or, they may be subsymbolically represented e.g. if a neural network was used.

Analysis of real instruments reveals that the tone of a single instrument can vary greatly when pitch is changed, or with changes in the volume of the playing. Therefore, the challenge in gross timbre identification is to identify the common features of the sound that identify a certain instrument, despite the large variations in tone that can be produced.

2.3 Timbre Analysis for Adjectival Timbre

A different body of work focuses on the concept of adjectival timbre. Here, the focus is not on studying the sound of an instrument as a whole, but on looking at individual generic characteristics of sounds such as brightness, harshness, or thickness. Early work on this was carried out by Grey [10], who identified some features of a synthesis algorithm which correlate strongly with timbral characteristics.

There are many studies in the field of psychoacoustics where experiments have been carried out to identify salient perceptual parameters of timbre. This experiments have usually taken the form of listening tests where volunteers have produced verbal descriptions of sounds, and the results are analysed to find correlations in the language used. This is useful as it identifies adjectives that are important for describing sounds, and this could form the basis for the types of perceptual features we might aim to control in the synthesiser program we are developing. However, these psychoacoustic experiments by themselves are not enough in order to synthesise the given perceptual features, since we also need to find a correlation of an adjective with certain spectral and temporal features of a sound; then more specifically with the parameters within a specific synthesis algorithm that give rise to those timbres or timbral changes.

The *SeaWave* project [4] made some progress in finding these correlations. Certain spectral and temporal features of starting sounds where modified, and the perceived changes in timbre where recorded. Some correlations where found and these were used to develop a synthesis system where certain timbral features such as resonance, clarity, or warmth could be controlled. The number of adjectives that where available to user to control the sound where limited, suggesting that more a much more extensive study of synthesis parameters and their perceptual correlates is needed.

It is interesting to note that while machine learning techniques have been used for automated classification of different instruments, it does not appear that a general system has been developed for automatically identifying adjectives that describe a certain sound. The small amount of work that has been carried out in this area has focused on specific domains. For example a recent paper by Disley and Howard [11] is concerned with the automated classification of timbral characteristics of pipe organ stops. It does not appear that any work has been carried out on automated classification of timbral *differences* between pairs of sounds.

2.4 Synthesis of Timbre

The most limited range of work to date has been on the automated *synthesis* of timbres or timbral changes.

Some work has been done on the automated synthesis of gross timbre. Clearly it is not possible to synthesise gross timbre from just words, but machine learning methods can be applied to learn synthesis parameters for a particular instrumental sound. In these cases the learning is guided either by interaction with a human [12, 13] or by the comparison of spectral features between the synthesized instrument-candidates and recordings of real instruments [14].

Of greater interest the this project is the automated synthesis of adjectival timbre. There are two basic concepts: associating adjectives/adverbs and classifications with timbres ('wooden', 'bright'), and words which are describe characteristics that sit on a timbral continuum ('can you play less reedily please?', 'let's have some more bounce'). A preliminary attempt to create a dictionary of such timbre-words, and to group them into classes, was attempted by Etherington and Punch [4].

A small amount of work has attempted to do automated synthesis of sounds from timbral descriptions. The *SeaWave* system [4] is based on a number of listening experiments which attempt to match specific sets of transformations of sound signals with words that describe those transformations. This works well up to a point; however the transformations required to achieve many kinds of transformations are likely to be complex, requiring more than simply the increase or decrease of a couple of synthesis parameters; and also they will typically be dependent on the starting sound.

Another attempt to generate quasi-timbral aspects of sound is given by Miranda [1]. He made use of machine learning methods to deduce correlations between parameters that could be used in synthesis and their perceived effects. This was then used to build up a database of matches between descriptive terms and characteristics which are used in synthesis; when the user requests a sound these characteristics are looked up in the database and a synthesis algorithm called with these characteristics. This provides a powerful methodology for generating sounds *ex nihilo*; however it was not applied to transforming existing sounds.

Since these two groundbreaking pieces of work, there appears to be no further work on linking linguistic descriptions of adjectival timbre to synthesis.

3 Complex Mappings: A Challenge for Timbre Exploration

One of the main difficulties with synthesis of timbres from descriptions is the complex nature of the mapping from the parameter space of a synthesis algorithm to the space of sounds, and then to the space of features that are described when we hear sounds (a more general exploration of such complexities in AI is given by Sloman [15]). Typically, many different closed subsets in parameter space will map onto the same timbre adjectives. Furthermore, features of timbre are influenced by previous exposure. For example, we are familiar with 'wooden' and 'metallic' sounds, and believe these to be contrasted; however in a synthesis algorithm it is possible to realise sounds that are physically unrealistic, e.g. sounds which are 'between' wooden and metallic, or which have both such characteristics.

This complexity contrasts with, say, loudness, where the mapping from the parameter (amplitude) to the perceptual effect (loudness) is straightforward. This presents a challenging problem for interfaces for timbre [2]; the timbral equivalent of the volume knob or piano keyboard is not obvious, nor is it obvious that such an interface could exist.

4 Experiments Timbre Synthesis Via Machine Learning

So far in this paper we have discussed work on the automated *analysis* of timbre, and on the *synthesis* of timbre. However there has been little progress in combining the results of these two approaches. in the remainder of this paper we present experimental work which attempts to combine these two ideas.

4.1 Approaches

There are basically two approaches to this problem. One is an *analytic* approach, where we work out directly how changes in the production of a sound lead to changes in the perceived timbral properties of the sound. The second, which we have used in our work, is a *machine learning* approach, where we take many examples of sounds, associate human-written metadata about timbre with them, and then apply machine learning methods [16] to create the relevant mappings.

An initial, somewhat naïve, perspective on this is to view it as being an *inverse problem.* That is, we take a set of sounds (recorded from acoustic instruments) that demonstrate the desired timbres (or timbral changes), and analyse what characteristics are common between the sounds that fit into an similar timbre-class. Then we apply analysis methods (e.g. spectral analysis) to these sounds to understand what characteristics of the sound 'cause' the different perceived timbral effects, and then apply these same characteristics to our desired sound to transform it in the same way.

However there are (at least!) two problems with this naïve model. Firstly, it is usually difficult to extract the characteristics that characterise a particular timbre or change of timbre. We can analyse sounds in many different ways, and not all of the characteristics that we can extract will be relevant to the production of a particular timbre. Even if we can eliminate some features by removing those characteristics that are common to sounds in various classes, there may be some arbitrary features that are irrelevant.

A second difficulty is found in the final stage of the process. Even if we can isolate such a characteristic, it is not easy to apply this to another sound: sometimes the characteristic can be difficult to express within a particular synthesis paradigm, and even if we can apply it, changing that characteristic in synthesis parameter space will not have the same effect in perceptual space. An additional problem of this kind is that, even when the changed sound is available within the synthesis algorithm being used, finding an appropriate change of parameters to effect the timbral change can be difficult.

4.2 System Overview

In our system we have tackled this problem in this indirect fashion, whilst avoiding the naïve approach of inverting the mapping. An overview of the program is given in figure 1.

An initial stage of the process consists of training a timbre classification algorithm. This takes a training set of many acoustic sound samples that have been hand-annotated with timbral metadata, and uses those to train a classifier which will sort sounds into relevant classes based on the timbre-words of interest. Some initial experiments with a neural network based classifier have proven promising.

The other main stage of the process consists of learning the parameter choice to feed into the synthesis algorithm. Sounds are generated using a synthesis algorithm (the algorithm used remains fixed throughout the process). The resultant sound files are then fed into the trained timbre classification algorithm, which outputs a measure of how strongly the sound fits into each timbral class.

This measure is then used as a quality measure (e.g. a fitness measure in a genetic algorithm [17]) in a machine learning algorithm. This could be used in a number of ways. One way would be to learn the values of certain input values in the synthesis algorithm which then remain fixed (in a similar fashion to [14, 18]). Another would be to learn which parameter changes (or characteristics characterised by covarying parameter changes, as in attribute construction in data mining [19]) are important in making particular perceived timbral changes.

The remainder of this section consists of a description of these two core components of the system.

4.3 Timbre Classification

In order for our learning system to be able to create a sound with the desired timbre, we need to be able to test the fitness of each solution that the system proposes. Of course, this needs to be an automated process, so our solution is to use a neural network capable of recognising certain timbral features in a sound.

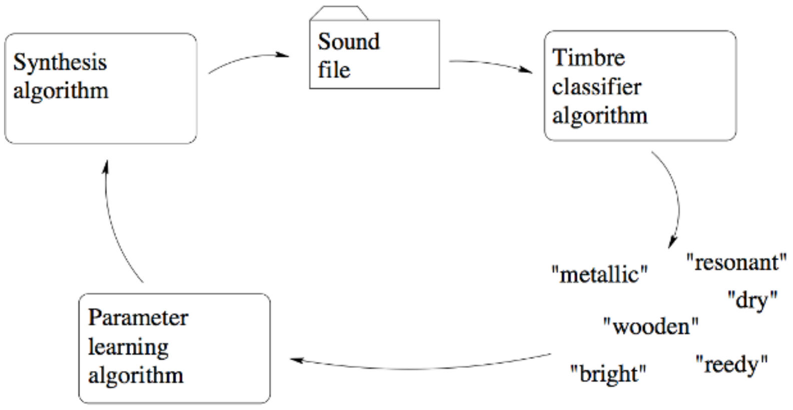
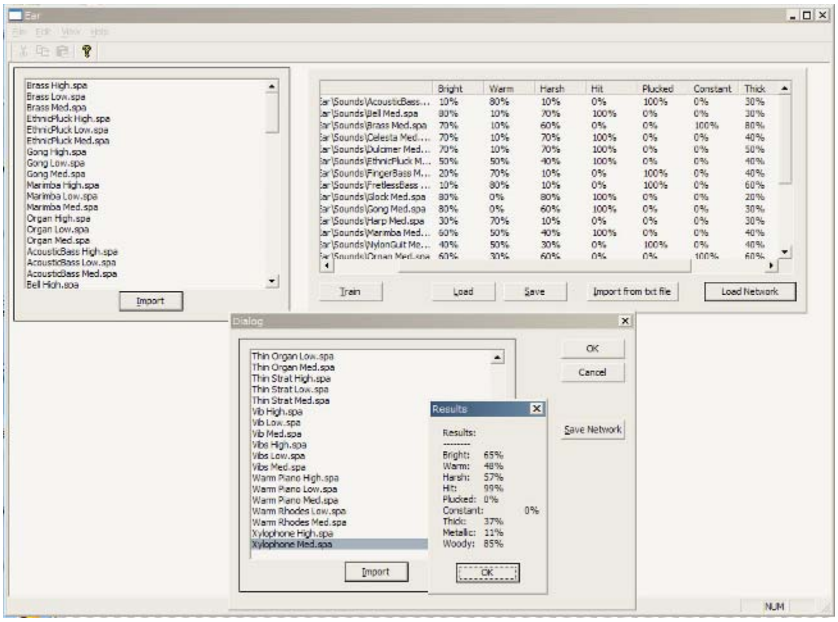


Fig. 1. Indirect learning of timbral change

Firstly, some pre-processing is carried out on the input sound wave in order to greatly reduce the complexity of the data, and therefore make it suitable for use as input to a neural network. Spectral analysis is carried out on the audio using an FFT, and the partials making up the sound are extracted from this, including the amplitude envelope and tuning information for each partial. From this data, a set of 20 inputs for the neural network is generated. Inputs 1-15 are the peak amplitude of each of the first 15 partials of the sound, which should describe the general 'colour' of the timbre. The next input is the average detuning of the partials, which describes how much the tuning of the partials differs from a precise harmonic series. The remaining inputs describe the overall amplitude envelope of the sound, and are attack time (time taken for the amplitude to reach its peak), decay time (time from the peak amplitude to the end of the note), and finally attack and decay slopes (rate of change in amplitude) which describe the general shape of the amplitude envelope.

The aim of the neural network is to map a set of inputs onto a set of values describing the timbral features present in the sound. In order to define the expected output of the network in our prototype, samples of notes from 30 (synthesised) instruments were collected, and 9 adjectives where chosen to describe the timbre of each instrument (bright, warm, harsh, thick, metallic, woody, hit, plucked, constant amplitude). Listening tests where carried out on each instrument sample, and values ranging from 0 to 1 were assigned indicating how well each adjective described the instrument (a value of 1 meaning the particular feature was strongly present in the sound, while 0 indicating that the adjective



Brass High.spa
Brass Low.spa
Brass Med.spa
EthnicPluck High.spa
EthnicPluck Low.spa
EthnicPluck Med.spa
Gong High.spa
Gong Low.spa
Gong Med.spa
Marimba High.spa
Marimba Low.spa
Marimba Med.spa
Organ High.spa
Organ Low.spa
Organ Med.spa
AcousticBass High.spa
AcousticBass Low.spa
AcousticBass Med.spa
Bell High.spa

	Bright	Warm	Harsh	Hit	Plucked	Constant	Thick
Ear\Sounds\AcousticBass...	10%	80%	10%	0%	100%	0%	30%
Ear\Sounds\Bell Med.spa	80%	10%	70%	100%	0%	0%	30%
Ear\Sounds\Brass Med.spa	70%	10%	60%	0%	0%	100%	80%
Ear\Sounds\Celesta Med....	70%	10%	70%	100%	0%	0%	40%
Ear\Sounds\Dulcimer Med...	70%	10%	70%	100%	0%	0%	50%
Ear\Sounds\EthnicPluck M...	50%	50%	40%	100%	0%	0%	40%
Ear\Sounds\FingerBass M...	20%	70%	10%	0%	100%	0%	40%
Ear\Sounds\FretlessBass ...	10%	80%	10%	0%	100%	0%	60%
Ear\Sounds\Glock Med.spa	80%	0%	80%	100%	0%	0%	20%
Ear\Sounds\Gong Med.spa	80%	0%	60%	100%	0%	0%	30%
Ear\Sounds\Harp Med.spa	30%	70%	10%	0%	0%	0%	30%
Ear\Sounds\Marimba Med...	60%	50%	40%	100%	0%	0%	40%
Ear\Sounds\NylonGuit Me...	40%	50%	30%	0%	100%	0%	40%
Ear\Sounds\Organ Med.spa	60%	30%	60%	0%	0%	100%	60%

Thin Organ Low.spa
Thin Organ Med.spa
Thin Strat High.spa
Thin Strat Low.spa
Thin Strat Med.spa
Vib High.spa
Vib Low.spa
Vib Med.spa
Vibs High.spa
Vibs Low.spa
Vibs Med.spa
Warm Piano High.spa
Warm Piano Low.spa
Warm Piano Med.spa
Warm Rhodes Low.spa
Warm Rhodes Med.spa
Xylophone High.spa
Xylophone Med.spa

```
Results:
---------
Bright:    65%
Warm:      48%
Harsh:     57%
Hit:       99%
Plucked:   0%
Constant:            0%
Thick:     37%
Metallic:  11%
Woody:     85%
```

Fig. 2. Timbre recognition process

did not apply to the sound at all). This work decided that our neural network would have 9 outputs onto which the inputs are mapped.

An application (figure 2) was developed that takes a list of sound files and their associated values for each adjective, carries out the necessary pre- processing to generate a set of inputs, and then trains a neural network to associate the correct adjectives with each sound's input data. A 3-layer back-propagation network was used, with 100 neurons per layer (this value was chosen empirically and gave reasonable training times as well as better generalisation than was achieved with smaller networks). Once the network is trained, the application allows the user to select an instrument sound that was not included in the training data, and run it through the system to classify its timbre.

4.4 Timbre Shaping

The second main part of the process is shaping the timbre, i.e. adjusting the parameters of the synthesis algorithm to produce the desired timbral characteristics. A screenshot of this program is shown in figure 3. The system uses an additive synthesis algorithm, using the parameters described in the previous section for synthesis rather than analysis.

In initial experiments, the system used a genetic algorithm (similar to [12]) to explore the parameter space and find a sound with the desired characteristics. The neural network timbre classifier was used to calculate a fitness for each solution in the population. However, tests of the program showed that the genetic algorithm had difficulty in finding suitable solutions, and did not show signs of converging towards a good solution. This is probably due to the sheer complexity of the mapping between synthesis parameters and the description of the timbre that they produce.

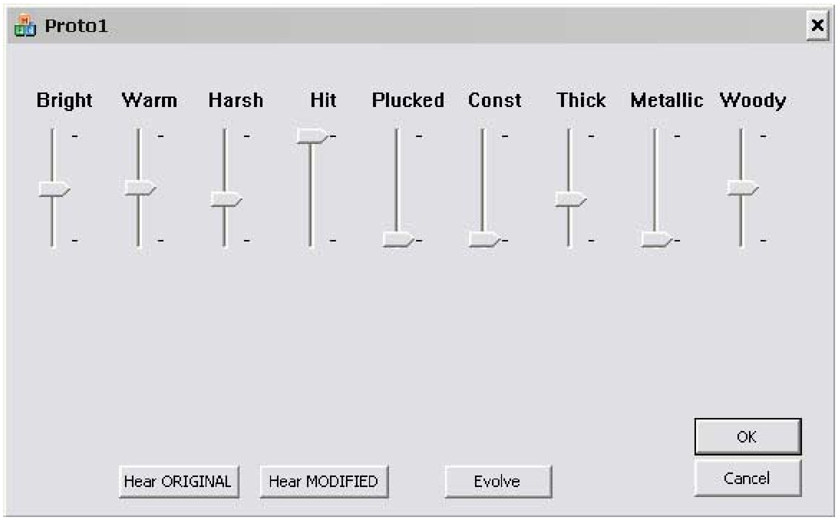


Fig. 3. The timbre shaping program

We have developed an alternative algorithm that is much more successful at finding the synthesis parameters for the desired timbre. It is designed to make use of the information that stored in the timbre classification neural network in order to search through the synthesis parameter space. The algorithm is similar to the back-propagation method used in training neural networks. Firstly, we take an arbitrary set of input values as a starting point and feed them into the neural network. These input values represent synthesis parameters. We run the network in order to obtain a set of results which represent a description of the timbre that would be produced. An error value for each output is then calculated, based on a comparison between the desired output and the actual output of the network. This error is then passed back through the network just as it is in the back-propagation algorithm, the only difference being that we do not actually modify any of the weights in the neuron connections. The error eventually propagates down to the inputs, therefore telling us how we need to adjust each input in order to obtain a better solution. The process then repeats until the overall error rate drops to an acceptable amount

There are clear reasons why this algorithm is more successful at finding a solution than the genetic algorithm. When using a genetic algorithm, each proposed solution is given a single fitness value which reflects how well it solves the problem. This means that we have no way of knowing how good each parameter in the solution is, since we can only judge the solution as a whole. Our algorithm however, gives us a separate error value for each parameter that makes up the solution, allowing us to move towards a good solution more effectively. Our algorithm makes use of the knowledge about timbre that is contained in the neural network, whereas with the genetic algorithm the genetic operators did not have sufficient power in combining and making small changes to timbral characteristics.

5 Results

5.1 Results of the Timbre-Classification Algorithm

The results of the timbre recognition process are presented in table 1. This shows a comparison between the timbral characteristics of five sounds, classified by a list of adjectives and a value indicated how strongly each characteristic was detected in the sound, by both a human listener and the neural network.

Early results from our timbre classification system are encouraging. In the experiment, a single human listener first assigned values to describe the timbre of the five test sounds, then the neural network was used to obtain a classification. The test sounds of course had not been used as part of the training set for the network. The results table shows that the prototype at this stage generally works well. There is evidence that the system is extracting common timbral features across different types of instrument sounds. It is particularly successful in detecting harshness, or sounds that are 'woody', or sounds that are hit. Unsurprisingly, it has trouble distinguishing between hit or plucked instruments, which is to be expected since the network's input data contains no information about

Table 1. The table first shows the expected value from a user listening test, followed by the neural network's actual answer in bold. A value of 1.0 indicates that a feature is strongly present in the sound, whereas a value of 0.0 indicates that the feature is absent.

Instrument	Bright	Warm	Harsh	Thick	Metallic	Woody	Hit	Plucked	Constant Amplitude
Vibraphone	0.6 **0.6**	0.5 **0.8**	0.4 **0.4**	0.4 **0.3**	0.5 **0.1**	0.3 **0.2**	1.0 **1.0**	0.0 **0.0**	0.0 **0.0**
Elec. Guitar	0.7 **0.3**	0.2 **0.6**	0.7 **0.7**	0.2 **0.4**	0.4 **0.2**	0.1 **0.3**	0.0 **0.6**	1.0 **0.4**	0.0 **0.0**
Piano	0.6 **0.7**	0.5 **0.4**	0.1 **0.3**	0.6 **0.6**	0.2 **0.0**	0.3 **0.2**	1.0 **1.0**	0.0 **0.0**	0.0 **0.0**
Xylophone	0.8 **0.7**	0.3 **0.5**	0.7 **0.6**	0.1 **0.4**	0.0 **0.0**	0.8 **0.9**	1.0 **1.0**	0.0 **0.0**	0.0 **0.0**
Elec. Piano	0.5 **0.2**	0.5 **0.9**	0.2 **0.1**	0.4 **0.1**	0.2 **0.1**	0.2 **0.2**	1.0 **1.0**	0.0 **0.0**	0.0 **0.0**

the spectrum of the sound's attack portion, which is known to be significant in recognising sound sources.

5.2 Results of the Timbre Shaping Algorithm

Some of the results from the timbre-shaping process can be heard at
`http://www.cs.kent.ac.uk/people/staff/cgj/research/evoMusArt2006/evoMusArt2006.html`

6 Future Directions

There are many future directions for this work. Some of these are concerned with timbre recognition, for example using ear-like preprocessing (as in [20]) of sound to generate the inputs to the neural network. We will also carry out more extensive human trials of the timbre-recognition experiments.

There are many future directions beyond this. A major limitation of many attempts at automated synthesis of timbre or timbre change is that the learning has been applied to a single sound. Future work will focus on learning transformations of synthesizer parameter space with the aim of finding transformations that will apply to many different sounds.

References

1. Eduardo Reck Miranda. An artificial intelligence approach to sound design. *Computer Music Journal*, 19(2):59–75, 1995.
2. Allan Seago, Simon Holland, and Paul Mulholland. A critical analysis of synthesizer user interfaces for timbre. In *Proceedings of the XVIIIrd British HCI Group Annual Conference*. Springer, 2004.

3. Trevor Wishart. *On Sonic Art.* Harwood Academic Publishers, 1996. second edition, revised by Simon Emmerson; first edition 1985.
4. Russ Etherington and Bill Punch. SeaWave: A system for musical timbre description. *Computer Music Journal*, 18(1):30–39, 1994.
5. Trevor Wishart. *Audible Design.* Orpheus the Pantomime, 1994.
6. Rosemary A. Fitzgerald and Adam T. Lindsay. Tying semantic labels to computational descriptors of similar timbres. In *Proceedings of Sound and Music Computing 2004*, 2004.
7. David M. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, 3(2), 1979.
8. S. McAdams, S. Winsberg, S. Donnadieu, G de Soete, and J. Krimphoff. Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes. *Psychological Research*, 58:177–192, 1995.
9. Boźena Kostek. *Soft Computing in Acoustics.* Physica-Verlag, 1999.
10. John M. Grey. *An Exploration of Musical Timbre.* PhD thesis, Stanford University, Department of Music, 1975.
11. A.C. Disley and D.M. Howard. Spectral correlates of timbral semantics relating to the pipe organ. *Speech, Music and Hearing*, 46, 2004.
12. Colin G. Johnson. Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In Geraint A. Wiggins, editor, *Proceedings of the AISB Workshop on Artificial Intelligence and Musical Creativity, Edinburgh*, 1999.
13. James McDermott, Niall J.L. Griffith, and Michael O'Neill. Toward user-directed evolution of sound synthesis parameters. In Rothlauf et al., editor, *Applications of Evolutionary Computing*, pages 517–526. Springer, 2005.
14. Jennifer Yuen and Andrew Horner. Hybrid sampling-wavetable synthesis with genetic algorithms. *Journal of the Audio Engineering Society*, 45(5):316–330, 1997.
15. Aaron Sloman. Exploring design space and niche space. In *5th Scandinavian Conference on AI.* IOS Press, 1995.
16. Thomas Mitchell. *Machine Learning.* McGraw-Hill, 1997.
17. Melanie Mitchell. *An Introduction to Genetic Algorithms.* Series in Complex Adaptive Systems. Bradford Books/MIT Press, 1996.
18. Janne Riionheimo and Vesa Välimäki. Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation. *EURASIP Journal on Applied Signal Processing*, 8:791–805, 2003.
19. Alex A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer, 2002.
20. David M. Howard and Andy M. Tyrrell. Psychoacoustically informed spectrography and timbre. *Organised Sound*, 2(2):65–76, 1997.

Modelling Expressive Performance: A Regression Tree Approach Based on Strongly Typed Genetic Programming

Amaury Hazan[1], Rafael Ramirez[1], Esteban Maestre[1], Alfonso Perez[1], and Antonio Pertusa[2]

[1] Music Technology Group, Pompeu Fabra University, Ocata 1, Barcelona 08003, Spain
ahazan@iua.upf.es, rafael@iua.upf.es, emaestre@iua.upf.es, aperez@iua.upf.es

[2] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Alicante, Spain
pertusa@dlsi.ua.es

Abstract. This paper presents a novel Strongly-Typed Genetic Programming approach for building Regression Trees in order to model expressive music performance. The approach consists of inducing a Regression Tree model from training data (monophonic recordings of Jazz standards) for transforming an inexpressive melody into an expressive one. The work presented in this paper is an extension of [1], where we induced general expressive performance rules explaining part of the training examples. Here, the emphasis is on inducing a *generative* model (i.e. a model capable of generating expressive performances) which covers all the training examples. We present our evolutionary approach for a one-dimensional regression task: the performed note duration ratio prediction. We then show the encouraging results of experiments with Jazz musical material, and sketch the milestones which will enable the system to generate expressive music performance in a broader sense.

1 Background

1.1 The Expressive Performance Modelling Problem

Modelling expressive music performance is one of the most challenging aspects of computer music. The focus of this work is to study how skilled musicians (here a Jazz saxophone player) express and communicate their view of the musical and emotional content of musical pieces by introducing deviations and changes of various parameters. The expressive performance modelling problem can be stated as follows. We define an expressive performance database B that consists of a set of pairs (S_i, E_i), where S_i is a set of melodic features of a given score note N_i, and E_i is a set of acoustic features describing the expressive transformations applied to note N_i. The problem is to find a model M that will minimise the error $Err(M(S_i)$, Ei) between the prediction $M(S_i)$ and the actual expressive transformation E_i for all the pairs (S_i, E_i) in B. Typically S_i is a set of features describing the melodic context of note N_i, such as its melodic and rhythmic intervals with its neighbours or its metrical position in the bar. On the other hand, E_i contain local timing and overall energy information of the performed note, but can also contain finer-grain information about some intra-note features (e.g. energy envelope shape, as studied in [2], or pitch envelope).

1.2 Regression Trees for Expressive Performance Modelling

In the past, we have studied expressive deviations on note duration, note onset and note energy [3]. We have used this study as the basis of an inductive content-based transformation system for performing expressive transformation on musical phrases. Among the generative models we induced, Regression Trees and Model Trees (an extension of the former) showed the best accuracy.

Regression Trees, are widely used in pattern recognition tasks, each non-leaf node in a Regression Tree performs a test on a particular input value (e.g. inequality with a constant), while each non-leaf node is a number representing the predicted value. By using a succession of IF-THEN-ELSE rules, Regression Trees iteratively split the set of training examples into subsets where the prediction can be achieved with increasing accuracy. The resulting structure is a coherent set of mutually-excluding rules which represents the training set in a hierarchical way. Figure 1 shows an example of a simple Regression Tree. This tree performs tests on 2 different features and returns a numerical prediction at its leaves. Whether this tree can produce accurate predictions when processing unseen data depends directly on the generalisation power of the model. To ensure good generalisation capability, tree induction algorithms, such as C4.5 [4] and M5 [5], provide pruning operations that rely on statistical analysis on the set of training examples.

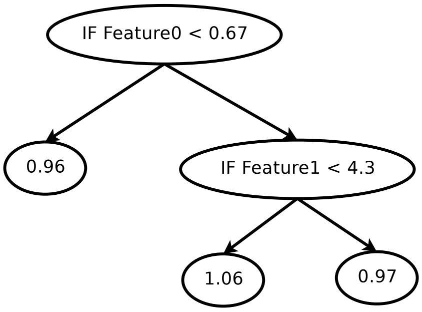


Fig. 1. A simple Regression Tree example. Tests are performed on 2 different features, namely Feature0 and Feature1. If a test succeeds, the left-side children is executed, while if the test fails, the right-side children is executed. When reaching a leaf, a numerical prediction is returned.

1.3 Evaluation Methodology for Expressive Transformation Data

We would like to add in the building process an evaluation of the model when predicting expressive transformations of musical excerpts in different musical situations. An example is to measure the system's ability to predict transformations of a given musical fragment at a given tempo when the model was built using training data based on recordings of the same tune but played slightly faster. Another example would be to compare the system behaviour when providing predictions on two dissimilar tunes when it was trained using data representing only one of these. In this work, we intend to evaluate the ability of our model to solve expressive musical performance situations considering these aspects.

1.4 An Evolutionary Computation Perspective

Most of tree induction algorithms are *greedy*, e.g. trees are induced top-down, a node at a time. According to [6], several works pointed out the inadequacy of greedy induction for difficult concepts. An alternative is to use Evolutionary Computation techniques, and especially Genetic Programming to evolve Regression Trees. By combining the high-level representation of computer programs and the near-optimal efficiency of learning with the parallel processing of several potential solutions, the Genetic Programming framework is an interesting direction for building such models. Koza [7] shows that this method has been successfully applied to concept learning and empirical discovery tasks.

Background on Genetic Programming
Genetic Algorithms (GA) can be seen as a general optimisation method that searches a large space of candidate hypotheses seeking one that perform best according to a fitness function. As presented in [8], GA transform a population of individuals, each with an associated value of fitness (i.e. ability to solve the problem), into a new generation of the population. The new generation is obtained using the Darwinian principles of survival and reproduction of the fittest and analogs of naturally occurring genetic operations such as crossover and mutation. In [7], Koza presents the Genetic Programming (GP) extension. While in the GA framework individuals are typically represented as binary or float strings, in the GP paradigm the structures undergoing adaptation are hierarchical computer programs of dynamically varying size and structure. These computer programs are represented as trees, in which a node represents a function, while each leaf is a terminal (e.g. input of the computer program).

Strongly Typed Genetic Programming
According to Montana [9], in standard GP, there is no way to restrict the programs it generates to those where the functions operate on appropriate data types. When the programs manipulate multiple data types and contain functions designed to operate on particular data types, this can lead to unnecessarily large search times and/or unnecessarily poor generalisation performance. This analysis is crucial in the context of building a Regression Tree model for expressive music performance. Indeed, tests operate on input variables

of the program, that is, inputs are compared to inputs which appear in the training set. On the other hand, leaves contain predictions which should reflect the distribution of outputs E_i in the training set. Although both inputs and predictions of our model can be coded as float values, building a program that performs direct comparisons between an input value, and an output value (i.e. a melodic feature in S_i and an expressive feature in E_i), would produce ineffective computations, and a considerable effort would be spent in order to generate individuals that reasonably fit the training examples. Montana presented an enhanced version of GP called Strongly Typed Genetic Programming (STGP) which enforces data type constraints. Using this approach, it is possible to specify what type of data a given GP primitive can accept as argument (e.g. type of input), or can return (e.g. output). To the best of our knowledge, this work presents a novel approach for building Regression Trees using STGP.

In section 2, we describe how to represent the problem and implement the structural constraints of Regression Trees using STGP types. Preliminary results in predicting one-dimensional expressive transformations of musical material are then presented in 3. Finally, Section 4 draws some conclusions and future work.

2 An Approach for Building a STGP Regression Tree Music Performance Model

2.1 Performance Training Data

Each score note N_i in the performance database B is annotated with a number of attributes representing both properties of the note itself and some aspects of the local context in which the note appears. Information about intrinsic properties of the note include the note duration and the note metrical position, while information about its context include the duration of the previous and following notes, and the extension and direction of the intervals between the note and the previous and following notes. Thus, each S_i in B contains six features sumarising this melodic information. In the last section, we will discuss the perspective of generating new melodic representations for building expressive models. Each note N_i is associated to a set of acoustic features E_i describing how the note was played by the performer. This information was obtained by applying a segmentation algorithm on the audio recordings and then performing an alignment of the annotated audio with the score (see [10] for a detailed description of these two steps). In this work, we focus on the duration ratio of the performed note, i.e. ratio between the performed note duration and the score note duration. This is obviously a first step before considering an extended set of expressive features as presented in [2] and [3]. Because this work is preliminary, we limited our training set to a few examples of the expressive performance database we maintain, namely two versions of the excerpt *Body And Soul*, performed at 60 and 65 beats per minute, and two versions of *Like Someone In Love*, performed at 120 and 140 beats per minute.

2.2 Types and Primitives

Going back to Figure 1 we can see that the general structure of the Regression Tree is the following. Each node is a test comparing an input with a value that was generated analysing the training data. Each leaf is an output value containing the numerical prediction. Thus, both inputs and outputs of the program should have different types. We define 4 different types, namely *InputValue*, *FeatValue*, *RegValue* and *Bool*. The first three types represent floating-point values, while the latter type represents boolean values that will be used when performing tests. Now we can define the primitives that will be used to build our models. They are listed in Table. 1.

Table 1. STGP primitives used in this study. Each primitive is presented along with the number of arguments it handles, the type of each argument, and its return type. Note that primitives EFeatValue and ERegValue, which are used for constant generation, take no argument as input.

Primitive Name	Number of arguments	Arguments Type	Return Type
IF	3	1st: *Bool*, 2nd and 3rd: *RegValue*	*RegValue*
LT	2	1st: *InputValue*, 2nd *FeatValue*	*Bool*
EFeatValue	0	-	*FeatValue*
ERegValue	0	-	*RegValue*

The **IF** primitive tests whether its first argument of type *Bool* is true or false. If the test succeeds, its second argument is returned, otherwise its third argument is returned (both second and third arguments have a *RegValue* type). The **LT** primitive tests whether its first argument is lower than the second argument (The former has a *InputValue* type and the latter *FeatValue*). The primitive returns a *Bool* which value is true if the test succeeds, false otherwise. Given the definitions of our types, we can see that during the building process the output of the **LT** primitive will always be connected to the first argument of the **IF** primitive. Instead, we could have chosen to use a single primitive **IFLT** with 4 arguments (the first two being involved in *InputValue* typed comparison, and the last two being used to return a *RegValue*). However, we think that restricting the tests to inequalities would be a constraint for future developments. *EFeatValue* and *ERegValue* are zero-argument primitives. We use them for generating constants. The first one generates constants to be involved in inequality tests (primitive **LT**). In order to produce meaningful tests, values produced by *EInput* are likely to be the values appearing in the vectors S_i of the training set. To ensure this, we collected all the features in S_i in B. When creating a *EFeatValue* primitive, we choose randomly one of them. Similarly, *ERegValue* primitive generates constants of type *RegValue* that will form the numerical predictions returned by the model. It is desirable that these predictions respect

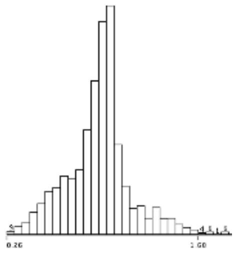


Fig. 2. Duration ratio distribution in the training data

approximately the statistical distribution of the vectors E_i in B. In this case, we focus on the single duration ratio in E_i. After having analysed the distribution of this value (see Figure 2.2), we decide to randomly generate *ERegValue* following a gaussian distribution fitting the training data.

2.3 Terminal Set

The elements of the terminal set are the inputs of the program. Each of the features of test vector S_i is associated to an input of the tree $IN_i, 0 \leq i \leq 5$. Terminals are of type *InputValue*, thus they will only feed primitives accepting arguments of this type (i.e. the LT primitive in the present case). Given the types, primitives, and terminals presented above, we are now able to build a Regression Tree structure in the STGP framework. Figure 3 shows how the example tree introduced in Figure 1 is represented.

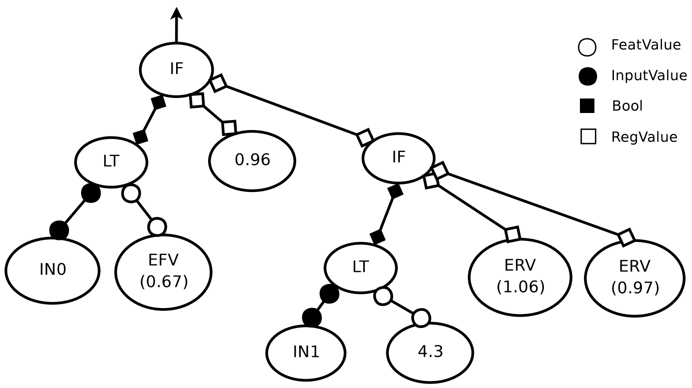


Fig. 3. STGP Regression Tree example involving two successive tests. Typed connections between primitives are appearing. Also, constants generated by zero-arguments primitives appear in parentheses. EFV stands for EFeatValue, while ERV stands for ERegValue. The arrow indicates the output point of the program.

2.4 Genetic Operators

The genetic operators we use are basically a refinement of the operators proposed by [7], and [9] in their Strongly Typed variant, where the operation is constrained to produce a type-consistent structure. Operators are Tree Crossover, where two individuals can swap subtrees, and Standard Mutation, where a subtree is replaced by a newly generated one. Additionally, Shrink mutation replaces a branch with one of its child nodes, and Swap mutation swaps two subtrees of an individual. Also, two floating-point mutation operators where added to process the randomly-generated constant returned by *EInput* and *ERegValue* primitives.

2.5 Fitness Evaluation

Individuals are evaluated on their ability to produce an output $M(S_i)$ that matches E_i for each note N_i of a musical fragment in the training set. Consequently, the fitness measure is the average error of the model $Err(M(S_i)$, Ei). We use the following fitness function:

$$f = \frac{1}{1 + RMSE} \tag{1}$$

where RMSE is the Root Mean Squared Error between training and predicted duration ratio, averaged over the notes of a musical fragment, itself averaged over all the training fragments. We are aware that this fitness measure is very general and that it does not catch in detail the behaviour of the model (e.g. evolution of the error note by note during the performance of a musical fragment). We have no guarantee that the best ranked individual will be the model that performs the best from the musical point of view. However this measure gives a first search direction during the evolution. Future developments will take advantage of perceptually-based fitness measures, but prior to this we have to address the issue of multi-dimensional prediction (see Sec. 4) in order to generate new melodies.

2.6 Evolutionary Settings and Implementation

We define the following evolutionary settings for the different runs. The population size is fixed to 200 individuals. The evolution stops if 500 generations have been processed or if the fitness reaches a value of 0.95. We use a generational replacement algorithm with a tournament selection. Crossover probability is set to 0.9, with a branch-crossover probability of 0.2, which means that crossovers are more likely to be performed on leaves, with the effect of redistributing the constants and terminals in the tree. The Standard mutation probability is set to 0.01, while the Swap mutation has been set to a higher value (0.1) in order to let a individual reorganise its feature space partition with more probability. Finally, Shrink mutation probability is set 0.05. The maximum tree depth has been set to 10, which could lead to the generation of very complex individuals (here we do not look for a compact model).

We build our system using Open Beagle framework [11], which is a C++ object oriented framework for Evolutionary Computation. Open Beagle was primarily designed for solving Genetic Programming tasks and includes STGP features such as the operators presented above.

3 Results and Evaluation in Different Musical Situations

In this section, we test the ability of our model in predicting expressive transformations given different training situations. Note that in this section we will not focus on the fitness of the individuals as defined in last section. Rather, we evaluate a model based on its error prediction. First, we address the problem of evaluating precision and generalisation capability of the induced model in the context of expressive music performance. In Figure. 4, we present the RMSE of the best-so-far model when predicting the note duration ratio of the four excerpts presented above. On the top, the model is only trained with one of the four fragments, namely *Body and Soul* played at 60 beats per minute, i.e, the fitness function is only based on this error. On the bottom, the model is trained using the four fragments. First we can see that excerpts that share the same

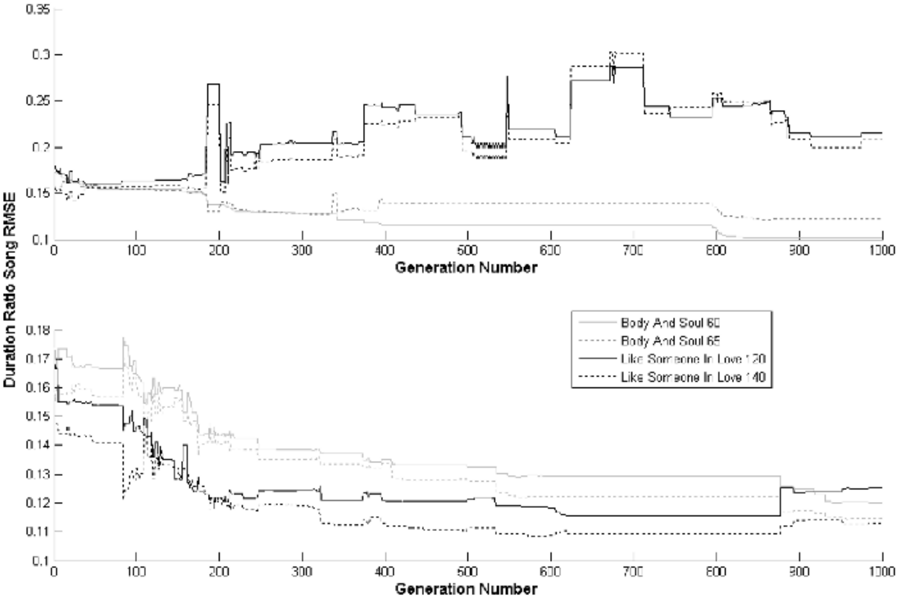


Fig. 4. Best-so-far individual RMSE for each of the target songs when the fitness takes into account only the *Body And Soul* (played at 60 beats per minute) prediction error (top), or when the fitness takes into account prediction the error of the four fragments (bottom). Gray solid (respectively dashed) line is the RMSE of *Body And Soul* played at 60 (respectively 65) beats per minute. Black solid (respectively dashed) line is the RMSE of *Like Someone In Love* played at 120 (respectively 140) beats per minute.

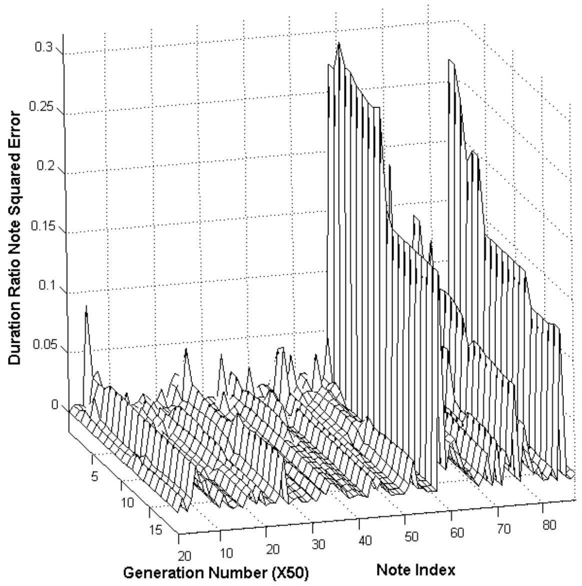


Fig. 5. Note by note Squared Error of the duration ratio during the evolution process

melodic representation evolve in a similar fashion. In the case of being trained with only one song (top of Figure. 4), the prediction error start to differentiate at generation 200. After this point, predictions concerning the training and similar fragments become better while predictions concerning dissimilar fragments become worse. On the other hand (bottom of Figure. 4), when all the fragments are used during training, the error tends to be minimised for the four excerpts. Some modifications of the model improve the prediction for all excerpts (e.g. generation 530), while others (e.g. generation 870) benefit only to excerpts sharing the same melodic features.

Figure. 5 shows the note by note Squared Error of the duration ratio during the evolution process. First generation model error appear on the back while last generation model error is on the front. We can see, that even if the overall error is minimised, some notes, such as note 61, 80, and 85 are modelled with less accuracy. This means that the best tree model does not cover these melodic situations appropriately. New specific branches should be created to perform tests on melodic features that represent theses melodic situations.

Finally, in Figure. 6, we perform an informal comparison between two models and the training data when predicting the duration ratio of each note the excerpt *Like Someone In Love* played at 140 beats per minute. The first model is the best-so-far STGP model obtained for this task (indicated in grey dotted line). The second model (grey dashed line) was obtained using a propositional greedy Regression Tree algorithm (see [3] for details), which is an accurate technique for the performance modelling task. The black solid line corresponds to the actual training data for this excerpt. We can see that both STGP and greedy Regression

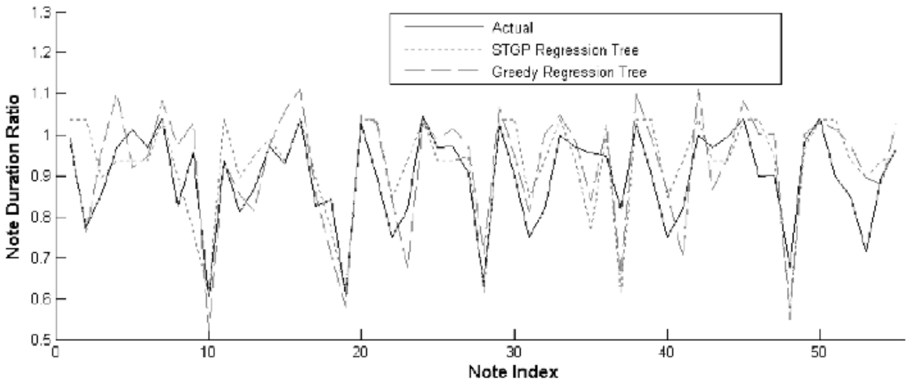


Fig. 6. Note by note duration ratio prediction for *Like Someone in Love* played at 140 beats per minute. Black solid line refers to the performance training data, grey dotted line refers to the best-so-far STGP model, and grey dashed line corresponds to a greedy Regression Tree model as presented in [3].

Tree models behave qualitatively well, and their mean prediction error is very similar. Thus, our approach to build Regression Tree performance models based on STGP is promising, although the results we present here are very preliminary.

4 Conclusion

We presented in this work a novel Strongly Typed Genetic Programming based approach for building Regression Trees in order to model expressive music performance. The Strongly Typed Genetic Programming framework has been introduced, along with the primitives, operators and settings that we apply to this particular task. Preliminary results show this technique to be competitive with greedy Regression Tree techniques for a one-dimensional regression problem: the prediction of the performed note duration ratio. We want to scale up our approach to the generation of expressive musical performances in a broader sense, and plan to work in the following directions:

– Predict more expressive features, such as onset and mean note energy deviation, which will enable the model to predict expressive melodies. This will be achieved by defining a new *RegValue* complex data type, along with an appropriate constant generator and operators. New fitness measures have to be defined and in order to assert the musical similarity between the model's output and the performer's transformations. We believe that the use of perceptually motivated fitness measures (e.g. [12]) instead of statistical errors (e.g. RMSE) can lead to substantial improvements of the accuracy of our models and make the difference with classical greedy techniques. Additionally, intra-note features are of particular interest in monophonic performances and will be considered. This will include energy, pitch, and timbrical features.

– Generate new melodic representations for our expressive performance database: as pointed out in Section. 2, the melodic representation we use is based on musical common sense but is not necessarily the best one. A particular drawback of this representation is that it only capture local context information, when several works stress that expressive music performance is a complex phenomenon which involves a multi-level representation of the music. An interesting work perspective is to devise a evolutionary melodic feature extractor that would coevolve with the performance model. [13] presents relevant ideas to achieve this.
– Increase the amount of training data. We are aware that our database is still small and that much more data is needed. New methods for the acquisition of performance data from polyphonic recordings (which would allow us to obtain data from commercial recordings) start to give interesting results. It is mandatory to build a representative performance database if we want to investigate thoroughly the generalisation (from both statistical and musical point of of view) power of the induced models. We will use statistical tests (e.g. 10-fold or one-song-out cross validation) to assess the performance of a given model. However we want to keep track of the sequences of events we are modelling, consequently we will avoid to use any technique which would lead to loose the temporal continuity in the data.
– Towards a model with state. A drawback in the model architecture is that it only bases its predictions on the melodic features of the note to be transformed, while it should also have access to the past predictions it returned. This last aspect is an important issue of the challenging expressive music performance problem.

References

1. Ramirez, R., Hazan, A.: Understanding expressive music performance using genetic algorithms. In: Third European Workshop on Evolutionary Music and Art, Lauzane, Switzerland (2005)
2. Ramirez, R., Hazan, A., Maestre, E.: Intra-note features prediction model for jazz saxophone performance. In: International Computer Music Conference, Barcelona, Spain (2005)
3. Ramirez, R., Hazan, A.: A tool for generating and explaining expressive music performances of monophonic jazz melodies. In: International Journal on Artificial Intelligence Tools (to appear). (2006)
4. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc. (1993)
5. Quinlan, J.R.: Learning with Continuous Classes. In: 5th Australian Joint Conference on Artificial Intelligence. (1992) 343–348
6. Murthy, S.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery **2** (1998) 345–389
7. Koza, J.: Genetic Programming: On the programming of Computers by means of natural Selection. MIT Press, Cambridge, MA, USA (1992)
8. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbour, MI (1975)

9. Montana, D.: Strongly typed genetic programming. Technical Report #7866, 10 Moulton Street, Cambridge, MA 02138, USA (1993)
10. Gómez, E., Grachten, M., Amatriain, X., Arcos, J.: Melodic characterization of monophonic recordings for expressive tempo transformations. In: Proceedings of Stockholm Music Acoustics Conference 2003, Stockholm, Sweden (2003)
11. Gagné, C., Parizeau, M.: Open beagle manual. technical report rt-lvsn-2003-01-v300-r1. Technical report, Laboratoire de Vision et Systèmes numérique, Université de Laval (2005)
12. Grachten, M., Arcos, J., López de Mántaras, R.: Melody retrieval using the implication/realization model. In: Online Proceedings of the 6th International Conference on Music Information Retrieval, London, UK (2005)
13. Conklin, D., Anagnostopoulou, C.: Representation and discovery of multiple viewpoint patterns. In: Proceedings of the International Computer Music Conference, La Havana, Cuba (2001) 479–485

Evolutionary Musique Concrète

Cristyn Magnus

Department of Music, University of California San Diego

Abstract. This paper describes a genetic algorithm that operates directly on time-domain waveforms to produce *musique concrète* compositions. The form of these compositions is derived from the evolutionary process itself. The aesthetic motivation is discussed and the results of the algorithm are described.

1 Background

Although electronic music experiments had been going on since the development of the telephone, a great breakthrough came in 1948, when Pierre Schaeffer broadcast his early studies in *musique concrète* on Radio-diffusion-Télévision Française[1]. *Musique Concrete* is a genre in which composers manipulate recordings of actual sounds rather than notes. Composers who use notes deal with abstract symbols that represent large categories of possible sounds; performances are unique interpretations of the symbols. A composer of *musique concrète* produces a definitive recording that is the piece; at performances, the recording is simply played. Techniques for composing with actual sounds give composers access to an extremely wide array of timbres—anything that could be recorded or brought out of a recording through manipulation. We are no longer restricted to pitches and rhythms that can be written using traditional western notational symbols.

Since the incorporation of recorded sounds is pervasive in contemporary electronic music, it is ironic that little attention has been given to developing techniques for manipulating recordings with genetic algorithms. Most research applying genetic algorithms to music has focused on symbolic music(see [2] for a review). Some research has broached the issue of timbre exploration through synthesis, but direct manipulation of recorded sounds has not been addressed. Johnson[3, 4] and Dahlstedt[5, 6] use interactive genetic algorithms to explore synthesis parameters. Horner, Beauchamp, and Packard[7] derive novel sounds with an interactive genetic algorithm that applies filtering and time-warping operations to populations of synthesized sounds. This comes closer to addressing recorded sounds, since filtering and time-warping need not be applied exclusively to synthesized sounds. These researchers all work to produce novel sounds that can be worked into later compositions. For a series of recent compositions, I have developed a technique that would allow me to use genetic algorithms to produce a series of pieces constructed from found sounds whose form would be derived from the evolutionary process.

2 A Genetic Algorithm That Operates on Time-Domain Waveforms

Since conventional genetic algorithms are meant to be applied to discrete symbols, applying them to sounds requires some modification. In my description of these changes, I will try to distinguish between practical choices that can be transferred to other musical projects and aesthetic choices that result in the characteristic sound of my pieces.

2.1 Representation

In a typical genetic algorithm, parameters are mapped onto genes and the ordered collection of genes forms a chromosome. Usually all chromosomes in the population have the same number of genes. My technique operates directly on digitized waveforms that can have arbitrary lengths. Each chromosome is a time-domain waveform. Using instantaneous samples as genes would be a bad idea: sexual reproduction would introduce clicks; mutation would introduce noise. So in my algorithm, there is no analysis and there are no discrete genes. Instead, a hybrid approach to genes is adopted. For the purpose of calculating fitness, samples are treated as genes. For the purpose of sexual reproduction and mutation, segments of waveform bounded by zero crossings are treated as genes.

Typically, a genetic algorithm runs for many generations. The initial population and any intervening generations are discarded; a representative member of the final population is chosen as the product of the algorithm. My algorithm produces a piece of music whose formal structure is a product of the evolutionary process. Each waveform produced by the algorithm becomes part of the final piece. A piece begins with the simultaneous playback of the initial waveform population. Whenever a waveform finishes playing, a new waveform is generated to take its place. The instant before a waveform's playback begins, its fitness is measured. Each waveform's playback volume is weighted by its fitness.

Since the output of the algorithm is a piece of music, choices regarding output representation are primarily aesthetic. If I wanted a piece with a different formal structure or simply a tool to generate sonic material to use elsewhere, I would make different choices.

2.2 Fitness

The choice of fitness function is primarily aesthetic. The purpose of a fitness function in my algorithm is simply to provide directionality for pieces produced by the algorithm and to determine the volume at which each waveform is played back. It is important that some waveforms be fitter than others for natural selection to take place. The fitness function is based on the correlation between waveforms in the population and a specified target waveform. Formally, this can be written as

$$Fitness = \frac{waveform \cdot target}{\|waveform\| \; \|target\|} b^n \tag{1}$$

where n is the number of times the waveform has reproduced and b is a parameter between 0 and 1. For $b = 0$, a waveform will never reproduce twice. For $b = 1$, a waveform's fitness is not reduced by reproduction. The b^n modifier is to encourage biodiversity (see §4.2).

Although a stripped down version of the algorithm can, under appropriate circumstances, produce sounds that can be recognized as imitations of the target waveform, this is not the compositional goal. The population is never expected to converge to some target. The biodiversity modifier lowers fitness each time a waveform becomes a parent to prevent the offspring of a handful of extremely fit individuals from dominating the population. In addition, the compositional framework for the piece (§3) has high-level control over the fitness function, which can change over the course of the piece.

2.3 Reproduction

Sexual reproduction is carried out by splicing genetic material from two individuals to produce one individual in the next generation. For each offspring, two parents are selected from the population. The probability that an individual will be selected as a parent is based on its fitness. Each parent is divided at some randomly selected crossover point. The location of the crossover point is adjusted to make sure it falls on a zero crossing. The first part of one parent is spliced to the last part of the other parent (figure 1). Because the crossover point is randomly selected and can be different for each parent, offspring can be arbitrarily short or potentially as long as the combined lengths of both parents.

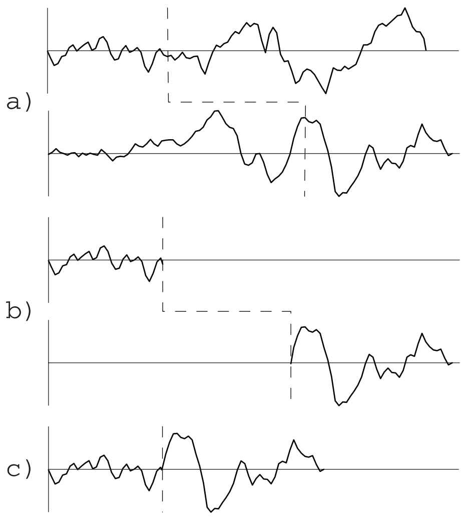


Fig. 1. a) Two parent waveforms (*solid line*) with their randomly selected crossover points (*dotted line*). b) The crossover point adjusted to fall on zero crossings. c) The child waveform.

I could have used a fixed crossover point, but I felt this was an opportunity to introduce rhythmic interest.

2.4 Mutation

Mutation occurs immediately after the offspring is produced, before its playback begins. Each segment of waveform between zero crossings has a slight probability of mutating. This mutated segment of waveform can include multiple zero crossings. Larger mutations are more perceptually relevant; that is, it is possible for a listener to identify mutated segments and sometimes even the type of mutation. Smaller mutations tend to denature the original sounds and produce waveforms that sound more like the target waveform.

A typical mutation function adds a random number to a gene. We can extend this concept to waveforms by adjusting a waveform's amplitude (figure 2a). This is done by selecting a random number and multiplying each sample of a waveform segment by that number. Another way of extending this concept is to raise each sample of a waveform segment by a power (figure 2b). To prevent the exponentiation from severely amplifying or attenuating the segment being mutated, each segment is normalized after exponentiation so that it retains its original maximum amplitude.

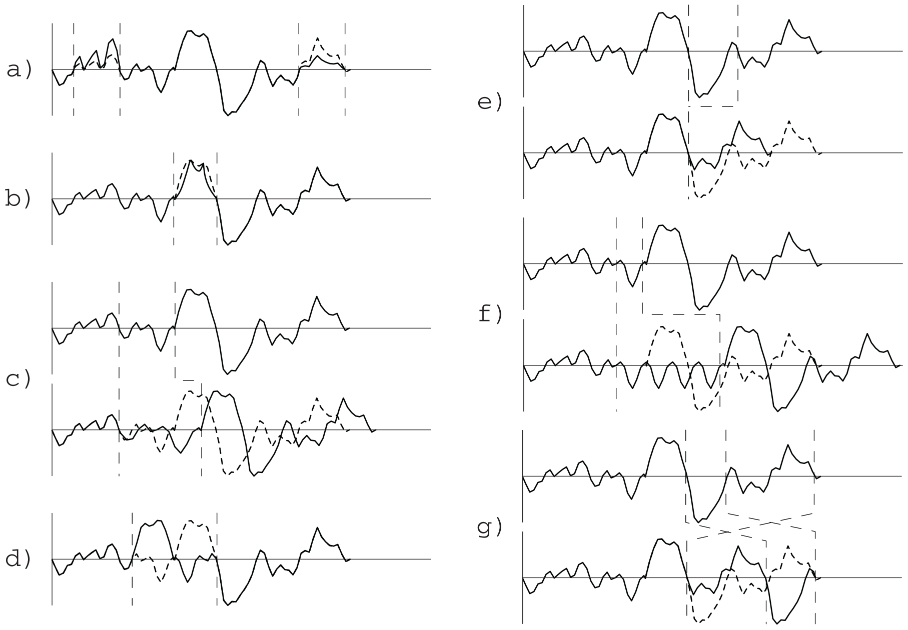


Fig. 2. Mutation operations showing the original waveform (*short dashes*) and the resultant waveform (*solid line*) with the mutation boundaries (*long dashes*): a) amplify b) exponentiate c) resample d) reverse e) remove f) repeat g) swap

We can think in terms of time rather than amplitude and resample a segment of waveform to lengthen it, making it lower in pitch, or to shorten it, raising its pitch (figure 2c).

Because mutation is applied to segments of waveform, rather than individual genes, we can draw inspiration from the types of errors that happen in actual gene transcription. Mutation functions can reverse a waveform segment (figure 2d), remove a waveform segment entirely (figure 2e), repeat a waveform segment a random number of times (figure 2f), or swap neighboring waveform segments (figure 2g).

3 Compositional Framework

In a single, unchanging environment, the algorithm described above would eventually converge to a local minimum where all individuals would have roughly the same length as the target waveform and would have acquired some of its amplitude envelope and frequency characteristics. To create formal compositional structure, I define a *world* in which the waveforms evolve. A world consists of multiple distinct environments that change over time.

For a given piece, the world will be characterized by some number of locations. These locations may be mapped spatially onto speakers. The *environment* at each location will initially be defined by some target waveform and some set of mutation probabilities. Immediately after an individual is created, it has a slight chance of moving to another location. If it migrates, it will pan from one speaker to the other over the course of its playback. It will be considered to be in the second location for its entire duration and will have its fitness determined there. It will be given the opportunity to reproduce in the second location, but not the first. In this way, sounds with new characteristics will enter each location, enhancing biodiversity.

The world will be characterized by probabilities of change. Both target waveform and mutation probabilities can change whenever a new waveform is created. There are two sorts of changes that environments can undergo. One is the slow drift that is seen in ice ages: these take place over an enormous amount of time from the perspective of individuals but happen many times over the evolution of a species. This is simulated by slowly cross-fading between two target waveforms. The other is the drastic change that results from catastrophic events, such as fire decimating a forest, causing it to be replaced by grassland. This is achieved by replacing the target waveform with a completely different waveform.

The changing environment prevents the population from strongly resembling the target waveform. The goal is to present the process, not draw attention to the underlying environment. Catastrophic environmental changes lead to musical surprises that reveal subsets of the population that were previously too unfit to be heard above the dominant sounds. Migration can have similar effects; it also increases biodiversity, which means there are always sounds in each location that can take advantage of the changing environment.

4 Results

4.1 General Description of Output

As evolution occurs, all of the waveforms in the population are written to a single sound file with each individual waveform weighted by its fitness. This weighting causes fit individuals to rise to prominence. Each time a waveform ends, a new individual is generated from the population. The new individual's playback begins immediately at the end of the waveform it replaces. Because the initial biodiversity is very high, the beginning of the output file is a wash of textures reminiscent of the timbres of the initial population. Within a few generations, a few fit individuals dominate the mix, causing a sound in which particular features of the initial population can be identified.

As evolution progresses, qualities of the initial population are preserved but are increasingly transformed through reproduction and mutation as the population takes on properties of the target waveform. The similarity to the target waveform depends on the type of mutation used, on the probability of mutation, and on the amount of time over which evolution occurs.

4.2 Biodiversity

In order for a piece to be musically interesting, biodiversity must be maintained. Since output is weighted by fitness, only fit sounds are heard distinctly. The truly musical moments occur when previously unfit sounds become fit, either through a changing environment or migration. Novel sounds bloom out of the sea of sounds and affect what is heard after they become fit.

4.3 Effects of Mutation on Output

Each type of mutation has a characteristic sound that can be readily heard if a population evolves with only that type of mutation. Amplification changes the population in two ways. The amplitude envelopes of individuals in the population tend towards the amplitude envelope of the target environment. Portions of individuals that are in phase with the target will be amplified, while portions that are out of phase will be attenuated. Exponentiation is very similar to amplification in its behavior, but it is much more invasive; it significantly alters the timbre of the waveform. Resampling allows pitch to become closer to the pitch of the target waveform.

The quality of the biologically inspired mutations (reverse, remove, repeat, swap) depends largely on the number of neighboring genes grouped for mutation. Application to large segments of the waveform leaves the waveform more recognizable but is less likely to add significantly to the fitness of the population. Given a population of individuals that are several seconds long, typically one or two lengthy mutations will audibly propagate to future generations over the course of a several minute piece. Application to very small segments of a waveform typically makes the original sound unrecognizable but is more likely to have a positive effect on fitness and be incorporated in the population.

When the biologically inspired mutations are applied to perceptibly large segments of a waveform, the function itself can be clearly identified. That is, the a listener can tell that a segment of a waveform has been reversed, removed, swapped with another segment, or repeated. When the grain size is fairly small, portions of the waveform tend to get shuffled around to more closely resemble the target waveform. Portions of a waveform that have been reversed tend to retain some quality that tells the listener that reversal has taken place, but the only biologically inspired mutation that has a significant fingerprint when applied to small segments of a waveform is repetition. Repetition creates pitch out of noisy segments of a waveform. When the grain size is small and the probability of mutation is high, repetition is effective at getting the population to denature to the point where the target environment can be recognized. For example, a listener unfamiliar with the target environment can identify the environment as a bell when listening to the evolution of a population of waveforms evolving with a bell as the target environment.[1]

4.4 Achieving Musical Results

Because the goal here is to make interesting music, rather than to attain a duplicate of some target sound file, I usually choose fairly small mutation probabilities and to apply mutations to fairly large segments of waveforms. This allows the sounds to be quite recognizable, even several minutes into the output file. The migration of individual waveforms from one environment to another and the ability of environments to change over time significantly contributes to the musicality of the output. I chose probabilities for both migration and environmental change that caused the trajectory of the piece to change every few minutes. This prevented the population from being dominated by the offspring of a few individuals and becoming monotonous.

5 Conclusion

I have used this algorithm to produce several pieces and an installation that have been performed and well received.[2] Many listeners have expressed surprise that the pieces were algorithmically generated with no composer intervention beyond setting initial conditions. This speaks to the algorithm's efficacy in producing novel and pleasing musical results. Depending on the source sounds and initial probabilities that I choose, I can generate very different pieces that share the characteristic sound of the algorithm. Over the course of a typical piece, sounds from the initial population slowly evolve. Rhythms change gradually; different sounds from the initial population rise to prominence at different points; and the piece has clear directionality, punctuated by occasional musical surprises.

[1] See `http://cmagnus.com/cmagnus/ga_results.shtml` for sample output.

[2] See `http://cmagnus.com/cmagnus/comp/gasketch.shtml` for a short piece.

References

1. Griffiths, P.: A Guide to Electronic Music. Thames and Hudson (1979)
2. Burton, A.R., Vladimirova, T.: Generation of musical sequences with genetic technique. Computer Music Journal **23**(4) (1999) 59–73
3. Johnson, C.G.: Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In Wiggins, G.A., ed.: Proceedings of the AISB Workshop on Articial Intelligence and Musical Creativity, Edinburgh (1999)
4. Johnson, C.G.: Exploring sound-space with interactive genetic algorithms. Leonardo **36**(1) (2003) 51–54
5. Dahlstedt, P.: Creating and exploring hute parameter spaces: Interactive evolution as a tool for sound composition. Proceedings of the International Computer Music Conference (2001) 235–242
6. Berry, R., Dahlstedt, P.: Artificial life: Why should musicians bother? Contemporary Music Review **22**(3) (2003) 57–67
7. Horner, A., Beachamp, J., Packard, N.: Timbre breeding. Proceedings of the International Computer Music Conference (1993) 396–398

A Connectionist Architecture for the Evolution of Rhythms

João M. Martins and Eduardo R. Miranda

Interdisciplinary Centre for Computer Music Research,
University of Plymouth, Plymouth,
Devon PL4 8AA, United Kingdom
{joao.martins, eduardo.miranda}@plymouth.ac.uk

Abstract. In this paper we propose the use of an interactive multi-agent system for the study of rhythm evolution. The aim of the model proposed here is to show to what extent new rhythms emerge from both the interaction between autonomous agents, and self-organisation of internal rhythmic representations. The agents' architecture includes connectionist models to process rhythmic information, by extracting, representing and classifying their compositional patterns. The internal models of the agents are then explained and tested. This architecture was developed to explore the evolution of rhythms in a society of virtual agents based upon imitation games, inspired by research on Language evolution.

1 Introduction

The early applications of evolutionary computation to music go back to 1991 with the works of Horner and Goldberg by applying genetic algorithms to thematic bridging [1]. Since then there have been many successful attempts to apply these techniques to music. For a discussion on the history and achievements genetic algorithms please refer to Gartland-Jones and Copley [2].

Neural Networks have also been used extensively in the context of music. There have been connectionist models for pitch perception, rhythm and metre perception, melody conduction and composition, many of them collected in Griffith and Todd's book [3].

Memetic theory, the cultural counterpart of biological evolution, was invented by Dawkins in 1979 [4], and postulates that culture is an evolutionary process evolving through the exchange, mutation and recombination of units of information that can be observed in different scales. Although the definition of a meme is still quite obscure, there have been some computational attempts to model the evolution of musical style according to this theory [5].

In the specific case of rhythm composition, we can find applications of evolutionary computation such as the Interactive Genetic Algorithm (IGA)from Horowitz [6] to breed drum measure loops, the CONGA system from Tokui and Iba [7] using genetic algorithms and genetic programming to generate rhythms which are evaluated by the user, and the creation of rhythms with cellular automata by Brown [8].

All these methods have been developed mainly with three applications in mind: Sound synthesis, composition, and musicology [9]. This paper focuses on the later; i.e., a framework for the study the evolution of music.

2 Imitation Games: Language and Music

Agent based modelling is a technique frequently seen in the A-Life context to study complex systems. The emergent behaviour of the system is observed when autonomous elements self-organise as a consequence of the interactions between each other and the environment. Regarding music, the applications of A-life models are described by Miranda and Todd [10]. The scope of the work presented on this paper considers a society of agents where rhythms are exchanged, processed and categorised with neural networks.

In the real world, there is no direct transposition of the knowledge between individuals, this meaning that it is not possible to copy all the information inside a person's brain and present it to another. In the case of language or a musical performance, this features get more accentuated as there is a strangulation in the channel and consequently in the amount of information that you are able to process. Although, is easy to exchange information in the computer without loss of data, for the purpose of simulation we need to find processing mechanisms and interaction schemes that can cope with this human limitation.

While some defend the innateness of Language and thus the role of genetic mutations in its evolution, Steels [11] defends that language corresponds to a Self-organising phenomena like the ones observed in chemical and biological processes. Furthermore language develops subject to big pressures of the environment, such as limited time for articulation of words, and acoustically adverse environments.

The same duality of opinions can arise on the musical side. The transmission media is the same as language, and music is also subject to the same kind of pressures, although not constrained to meanings and concepts. Werner and Todd [12] put emphasis on the role of mate selecting pressures for the evolution of repertoires, and the evaluation of the specimen fitness is made the according to the musical material. Miranda [13] explored the self-organising potential of agents' societies by furnishing the agents with motor and auditory skills and letting them evolve a shared repertoire of short sound sequences through imitation games .

Originally inspired by Wittgenstein [14], Luc Steels [15] proposed a model of imitation games for artificial agents. Bart de Boer [16] applied this game methodology to study the emergence of a coherent vowel system handling phono-articulatory parameters. Miranda [17] applied a slightly different version of the algorithm to develop intonations. Basically the game consists of one agent picking a random sound from its repertoire and the other agent trying to imitate it. Then feedback is given about the success of the imitation. On the basis of this feedback, the agents update their vowel repertoires.

Our approach differs from the applications previously presented in the sense that the judgement is made upon a system of internal categories of each of the agents and how the repertoire evolves in the continuous search to generate music that the other agent will recognise in his internal categories system.

In this paper we introduce the groundwork that characterises our approach; i.e., the connectionist nature of the agent's mechanism for representing rhythms.

3 Agents Architecture

We will present the architecture an agent containing two neural networks in cascade that receive a stream of rhythmic events as input and contain three output neurons that map these rhythms into a tridimensional space. For a comprehensive foundation on neural network theory please refer to Haykin's book [18].

Each agent is provided with a set of two neural networks: a SARDNET and a one layer Perceptron (Figs 2 and 5). The first one receives the stimulus sequentially from an input, encoded as a MIDI stream of rhythmic events, and generates an activation pattern corresponding to the agents perception of the type of event and its place in the sequence. The dynamics of this network is fully explained in Sec. 3.1. The pattern of activation from the Sardnet then becomes the input of the later network, the Perceptron, which generates three output values that enable the categorisation of the received sequences. The architecture and learning rules of the Perceptron are explained in Sec. 3.2.

The events are represented as vectors with three components. The first component defines the musical instrument (timbre), the second defines the loudness (velocity), and the third defines the value in milliseconds that the sound lasts (Inter-onset interval). These three dimensions correspond to human perceptual attributes with different scales in sensitivity and range. Modelling these differences in the learning algorithm was not part of the scope of this paper.

3.1 Sardnet

The SARDNET [19] is a self-organising neural network for sequence classification that was applied in phonology and recently it was also applied to simulations for evolving melodies [20]. This network is an extension of the original Self Organised Map (SOM) which is a neural network used for unsupervised learning developed by Kohonen [21]. The SOM has proven to be a powerful tool for many engineering applications and some of its variations have provided explanations for the organisation and development of the visual cortex [22].

The SOM is also called a competitive network or "winner-takes-all" net, since the node with largest input "wins" all the activation, which reflects on the possibility of updating that unit in order to become more similar to the input. The neighbouring units of the winning neuron are also updated according to a neighbourhood function that organises representations of similar stimuli in a topographically close manner.

In Fig. 1 we can see a diagram of a SOM with 16 units and one input. The dimension of the input vector determines the dimension of the weights vector of each unit. To determine which weight vector is the closest one to the input unit, the euclidean distance is calculated:

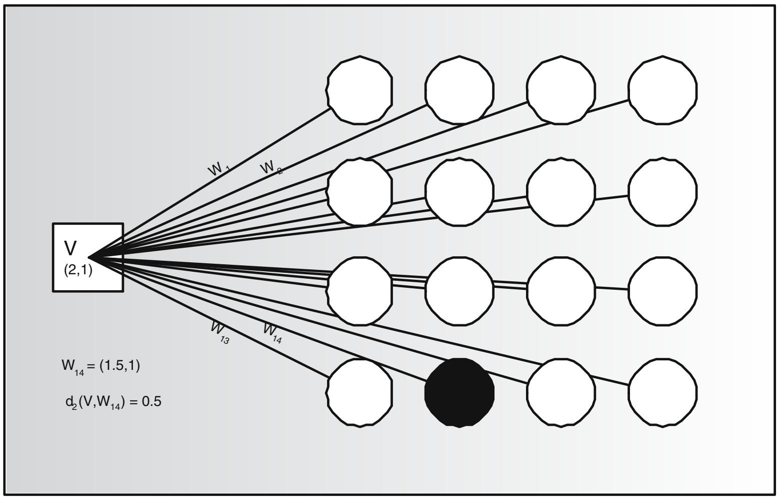


Fig. 1. Kohonen Self Organising Feature Map (SOM)

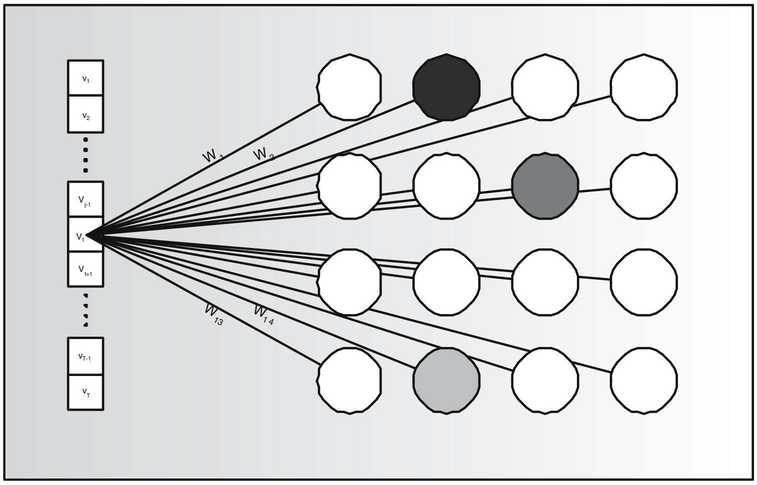


Fig. 2. SARDNET - Self-organising activation, retention and decay network

$$d_2(\mathbf{v}, \mathbf{w}) = \sqrt{\sum_{i=1}^{n} |v_i - w_i|^2} \tag{1}$$

The SARDNET keeps some essential features from the SOM, but adds two important features that enables us to deal with sequences of events. The first diverging characteristic is that the winning neuron is removed from subsequent competitions, and the second difference corresponds to holding the previous activations with a decay in each time step. The dynamics of SARDNET is shown on Fig. 2 where we can observe a the stream of events passing through the input and activating three units in sequence (W_{14}, W_7, W_2). The training algorithm for the SARDNET is shown on Tab. 1.

Table 1. The Sardnet training algorithm

INITIALIZATION: Clear all map nodes to zero
MAIN LOOP: While not end of sequence 1. Find inactive weight vector that best matches the input. 2. Assign 1.0 activation to that unit. 3. Adjust weight vectors of the nodes in the neighbourhood. 4. Exclude the winning unit from subsequent competitions. 5. Decrement activation values for all other active nodes.
RESULT: Sequence representation = activated nodes ordered by activation values.

Like the SOM, the SARDNET uses the Euclidean distance $d_2(w, v)$ from Eq. 1 to evaluate which is the weight that better matches the input. On step 3 of the algorithm the weight of the winning and the neighbourhood units are changed according to the standard rule of adaptation:

$$\Delta w_{jk} = \alpha(w_{jk,i} - v_i) \tag{2}$$

where α depends also on the distance to the winning unit, meaning its position in the neighbourhood. The neighbourhood function decreases as the map becomes more organised.

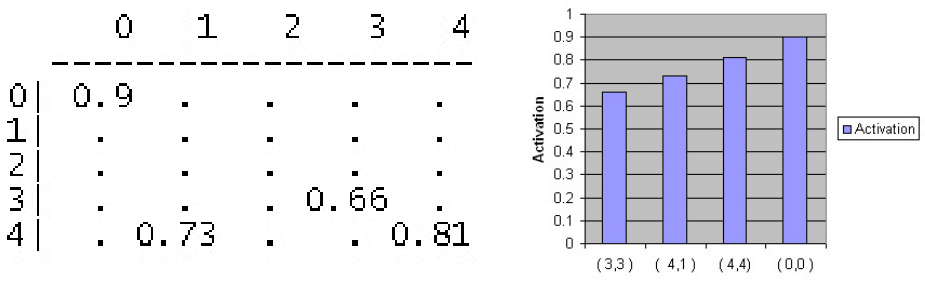
As in step 5 of the algorithm, all the active units are decayed proportionally to the decay parameter d,

$$\eta_{jk}(t+1) = d\eta_{jk}(t), \qquad 0 < d < 1 \tag{3}$$

In the following section we present the details of the Perceptron, the network that receives the activation patterns from the SARDNET, keeping the relevant information about this activation patterns across several sequences.

3.2 Perceptron

The Perceptron is a neuron-like learning network developed by Rosenblatt [23] which is a one layer feed-forward neural network with a set of inputs that are fully connected to an output layer. The outputs of Perceptrons are explicit functions of the inputs. Fig. 5 shows its architecture.



```
      0    1    2    3    4
   --------------------------
0|  0.9   .    .    .    .
1|   .    .    .    .    .
2|   .    .    .    .    .
3|   .    .    .  0.66   .
4|   .  0.73   .    .  0.81
```

Fig. 3. Activations from the output layer on in two different views

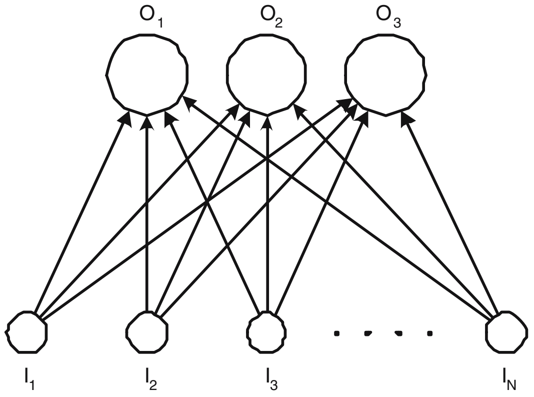


Fig. 4. Perceptron network

$$O_i = g(h_i) = g\left(\sum_k w'_{ik} I_k\right) \tag{4}$$

Eq. 4 is the propagation function of the Perceptron and $g(h)$ in Eq. 5 is the activation function computed by the units. In this case this function is a sigmoidal function,

$$O_i = g(h_i) = \frac{1}{1 + \exp(-h_i)} \tag{5}$$

The Perceptron uses the gradient descendant method to change the weights in order to adjust the test input to a given target.

$$\Delta w_{jk} = \eta * (T_k - O_k) I_j; \tag{6}$$

where η is the learning rate, T is the target value and $T_k - O_k$ is the corresponding error during the training phase.

The number of inputs of the Perceptron is the number of units of the SARD-NET. The number of output neurons is arbitrarily defined as being 3 to be able to visualise the results in a tridimensional grid. This output grid enables the categorisation of the input sequences.

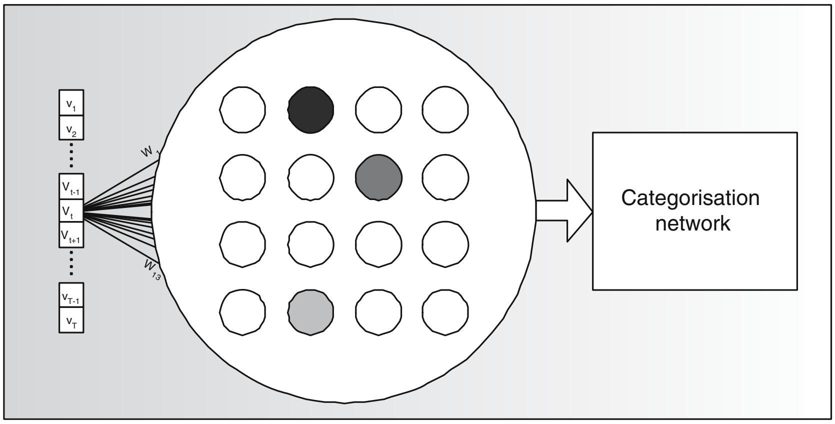


Fig. 5. Interaction diagram for the imitation game proposed

4 Analysis of the Agent

4.1 Sardnet

First we trained the Sardnet solely with prerecorded rhythms. We used a map with 50 elements, 10 in the length and 5 in the breadth, a learning rate of 0.1. The map was initialised with random weights in the range of -1 to 1. To perform the first organisation tasks the map was fed with 5 sequences of rhythms of latin music, each of them containing one or two instruments, very much like it would be if these were performed by other agents. After a couple of iterations a pattern of organisation could already be observed in the network, but the correspondent sequences extracted sounded extremely chaotic. After 50 iterations the rhythms start to sound organised as well, and the changes to the timbre of the instrument have the largest perceptual impact. This was expected to be so, as there is no discrimination in the organisation algorithm regarding the different weight components. Nevertheless, the organisation process is fine tuned enough to adapt perceptually perfectly to the incoming sequence after 80 iterations, and a learning musician is also expected to make timbre mistakes.

The graphs from Fig. 6 show the evolution of the third component of the weights (Inter-onset Intervals). The first graph shows the initial value of the weights, as explained above, the second shows the organisation process after 20 iterations, and the third shows the weights stabilised after 80 iterations. Fig. 6 d) shows the difference between the sums of the weights in two consecutive iterations, this being a measure of the stabilisation of the weights.

Previously it was stated that the SOM adapts its weights, not only for the winning elements, but also in its neighbourhood. In Fig. 7 it is shown the same organisation process but considering the neighbourhood change. The parameter σ controls the range of the the gaussian that changes the neighbourhood. By using an initial value of $\sigma = 2.97$ we can more rapidly capture the global charac-

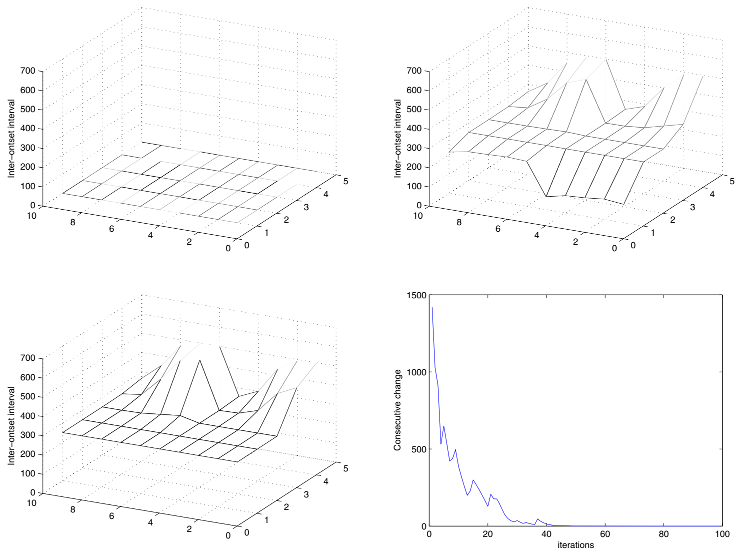


Fig. 6. Sarnet weight evolution without change of neighbourhood: a) Weight initialisation; b) After 20 iterations; c)After 80 iterations; d) Difference between the weights' sum in consecutive iterations

teristics of the input. It is necessary to reduce gradually this value in order not to destroy the representations of the events that occur less frequently. Comparing Figs. 6d) and 7d) we see that this procedure accelerates the convergence process.

One of the most important conclusions is that although it is possible to extract very similar sequences from both maps, the internal representation can be quite different, as can be seen from both Figs. 6 and 7 both trained with the same sequences.

4.2 Perceptron

The Perceptron's architecture is explained in Sec. 3.2. The Perceptron used for these experiments had 50 input units, that receive their values directly from the activations of the output layer of the Sardnet. These input units are fully connected to 3 output neurons enabling the mapping and categorisation of the input sequences into a tridimensional space of straightforward visualisation. We chose the first three activation layers of 50 elements corresponding to three rhythms fed previously to the Sardnet, and trained the Perceptron to respond to these patterns with three different targets, namely $[1, 0, 0]$, $[0, 1, 0]$ $[0, 0, 1]$. This process took 434 epochs to reach an error of categorisation of 10^{-3} as can be seen in Fig. 8 a). Each training patterns is marked with an (o) in the categorisation space (Fig. 8 b)). Later, we fed the perceptron with the last two rhythms and observed its activation marked with an (x). These were found to be much closer to the $[0, 1, 0]$ target, which interestingly correspond to the most similar pattern regarding the IOIs.

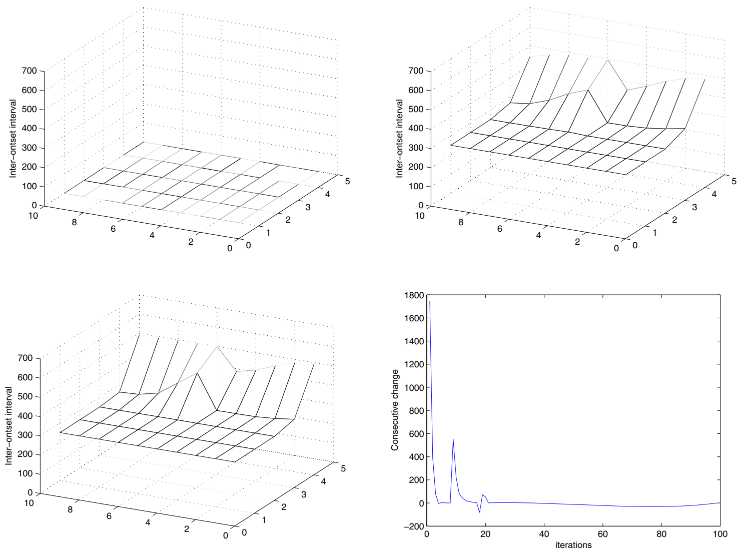


Fig. 7. Sarnet weight evolution with change of neighbourhood: a) Weight initialisation; b) After 20 iterations; c)After 80 iterations; d) Difference between the weights' sum in consecutive iterations

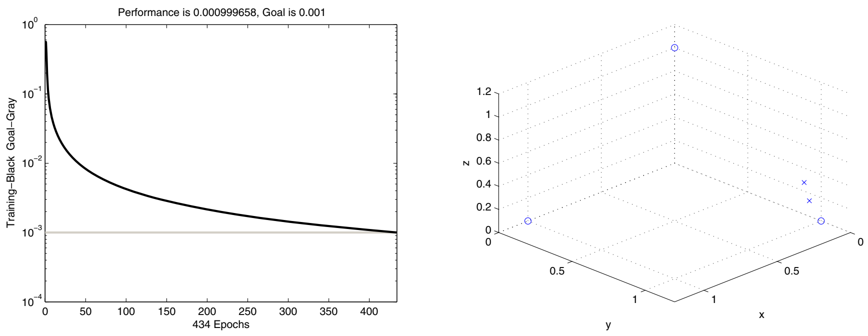


Fig. 8. a) Perceptron error in learning process; b) Categorisation space

5 Conclusion

With this paper we presented the architecture of an interactive virtual agent that is able to learn rhythms. The agent is composed of two neural networks that are able to learn the rhythms representation through self-organising processes. As it happens with humans, the agents always have different internal representations for the rhythms they listen to. Furthermore, the output of the networks categorises the incoming sequences and provides a measurement for the agents to judge how related are the listened rhythms. The rhythm representation allows for all types of rhythms to be encoded, considering event variables of Inter-onset interval, timbre and intensity. Several tests to the individual networks were made to show the potential to evolving rhythms and categories. We are now studying the results of number of simulations of imitation games where different rhythmic repertoires were evolved from scratch under a variety of different scenarios.

References

1. Horner, A., Goldberg, D.: Genetic algorithms and computer-assisted music composition. In: Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kauffman (1991)
2. Gratland-Jones, A., Copley, P.: The suitability of genetic algorithms for musical composition. Contemporary Music Review **22**(3) (2003) 43–55
3. Griffith, N., Todd, P.: Musical Networks. MIT-Press, Cambridge:USA (1999)
4. Blackmore, S.: The Meme Machine. Oxford University Press (1999)
5. Gimenes, M., Miranda, E.R., Johnson, C.: A memetic approach to the evolution of rhythms in a society of software agents. In: Proceedings of the 10th Brazilian Symposium of Musical Computation (SBCM), Belo Horizonte (Brazil) (2005)
6. Horowitz, D.: Generating rhythms with genetic algorithms. In Anderson, P., Warwick, K., eds.: Proceedings of the International Computer Music Conference, Aarhus(Denmark), International Computer Music Association (1994)
7. Tokui, N., Iba, H.: Music composition with interactive evolutionary computation. In: Proc. 3rd International Conf. on Generative Art, Milan, Italy (2000)
8. Brown, A.R.: Exploring rhythmic automata. In Rothlauf, F., et al., eds.: Proceedings of the 3rd European Workshop on Evolutionary Music and Art, Lausanne(Swizerland), Springer Verlag (2005)
9. Coutinho, E., Gimenes, M., Martins, J., Miranda, E.R.: Computational musicology: An artificial life approach. In: Proceedings of the 2nd Portuguese Workshop on Artificial Life and Evolutionary Algorithms Workshop, Covilhã(Portugal), Springer Verlag (2005)
10. Miranda, E., Todd, P.: A-life and musical composition: A brief survey. In: Proceedings of the IX Brazilian Symposium on Computer Music, Campinas,(Brazil) (2003)
11. Steels, L.: The synthetic modeling of language origins. Evolution of Communication **1**(1) (1997) 1–34
12. Werner, G., Todd, P.: Too many love songs: Sexual selection and the evolution of communication. In Husbands, P., Harvey, I., eds.: ECAL97, Cambridge, MA, MIT Press (1997) 434–443
13. Miranda, E.R.: Emergent sound repertoires in virtual societies. Computer Music Journal (MIT Press) **26**(2) (2002) 77–90
14. Wittgenstein, L.: Philosophical Investigations. Blackwell Publishers (1979)
15. Steels, L.: A self-organizing spatial vocabulary. Artificial Life **2**(3) (1995) 319–332
16. de Boer, B.: Self-Organisation in Vowel Systems. PhD thesis, Vrije Universiteit Brussel AI-lab (1999)
17. Miranda, E.R.: Mimetic development of intonation. In: Proceedings of the 2nd International Conference on Music and Artificial Intelligence (ICMAI 2002), Springer Verlag - Lecture Notes on Artificial Intelligence (2002)
18. Haykin, S.: Neural Networks. Prentice Hall, New Jersey:USA (1999)

19. James, D.L., Miikkulainen, R.: SARDNET: a self-organizing feature map for sequences. In Tesauro, G., Touretzky, D., Leen, T., eds.: Advances in Neural Information Processing Systems 7, Cambridge, MA, USA, MIT Press (1995) 577–84
20. Bosma, M.: Musicology in a virtual world: A bottom up approach to the study of musical evolution. Master's thesis, University of Groningen (2005)
21. Kohonen, T.: Self-Organizing Maps. Springer Series in Information Sciences. Springer-Verlag Berlin and Heidelberg GmbH & Co. K (1997)
22. Bednar, J.A., Miikkulainen, R.: Joint maps for orientation, eye, and direction preference in a self-organizing model of V1. Neurocomputing (in press) (2006)
23. Rosenblatt, F.: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Spartan Books, Washington (1962)

MovieGene: Evolutionary Video Production Based on Genetic Algorithms and Cinematic Properties

Nuno A.C. Henriques[1], Nuno Correia[1], Jônatas Manzolli[2], Luís Correia[3], and Teresa Chambel[3]

[1] New University of Lisbon, Caparica 2829-516, Portugal
nach@fct.unl.pt, nmc@di.fct.unl.pt
[2] Campinas State University, Campinas, Brazil
jonatas@nics.unicamp.br
[3] University of Lisbon, Lisbon 1749-016, Portugal
luis.correia@di.fc.ul.pt, tc@di.fc.ul.pt

Abstract. We propose a new multimedia authoring paradigm based on evolutionary computation, video annotation, and cinematic rules. New clips are produced in an evolving population through genetic transformations influenced by user choices, and regulated by cinematic techniques like montage and video editing. The evolutionary mechanisms, through the fitness function will condition how video sequences are retrieved and assembled, based on the video annotations. The system uses several descriptors, as genetic information, coded in an XML document following the MPEG-7 standard. With evolving video, the clips can be explored and discovered through emergent narratives and aesthetics in ways that inspire creativity and learning about the topics that are presented.

1 Objectives

The broader goal of this research is to find new ways of editing and producing multimedia documents. In our approach, the main objectives are: (1) To use Evolutionary Computation for the creation of a new paradigm for multimedia production; (2) To develop annotation mechanisms enabling fitness evaluation of a set of video clips; (3) To use the system in an interactive way, so that the evolutionary process may be affected by the user.

The system that we are proposing, MovieGene, is a multimedia production system, that uses genetic algorithms[1] and user selection, as a way to evolve a population of video segments. These segments are previously annotated with metadata (MPEG-7 [2] descriptors), that is used in the selection process. The user actions may influence the evolutionary process as a selection operator. Concepts inspired in video editing techniques are used to assemble the resulting videos.

2 Video Annotation, Metadata, and Evolving Video

Annotation should address both high level semantic information, entered by humans annotating the video, and also low level information, such as color histograms, obtained automatically. The relative start time and the duration (variable) of each segment, within a video document, is of interest to explore (evolve) and hence to code into the chromosome as annotations. The fitness function evaluates color similarities between genes of different chromosomes, using histograms in the `GoFGoPColor` descriptor. Text annotation is used in order to have semantic information, in the `FreeTextAnnotation` and `KeywordAnnotation` descriptors. The shot type is also used. We defined a set of video clips as the population, video annotations of each shot as the genotype, and the video editing process as the expression process. Video sequences (phenotype) are taken as expressions of video annotations (genotype). Each gene contains the full set of descriptors, for the characterization of a video segment (figure 1).

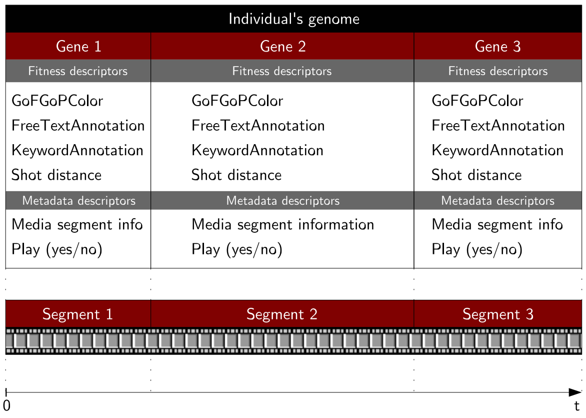


Fig. 1. Three atomic video segments (genes) example

3 Fitness Function

The fitness function f is formulated (equation 1) as the sum of all distances between individual and goal genes. For each gene (segment) value, the descriptors' distance weighted sum is calculated: the similarity matching of `GoFGoPColor` (C) descriptors of a segment and the specified goal colors; the `KeywordAnnotation` (K) similarity matching using a proposed algorithm for distance computation [3]; the `FreeTextAnnotation` (F) similarity matching using a developed [3] hybrid algorithm, that uses the Levenshtein[1] algorithm; and an *ad hoc* similarity matching function for the `ShotDistance` (S) proposed [3] descriptor.

[1] `http://www.nist.gov/dads/HTML/Levenshtein.html`

The equations for the distance measurement between each individual's descriptor, at some segment, and the goal (purpose to reach) are of the generic form $Z = d(Z_i, Z_{goal})$. The formula [2] for matching and measuring the distance between two distinct `GoFGoPColor` descriptors, G and G', is $C = \sum_n |G_n - G'_n|$, where $C = d(G, G')$. The number of coefficients for the color histogram is represented by n. As mentioned, the metrics for free text similarity $F = d(s_1, s_2)$ and for the keywords distance $K = d(s_1, s_2)$ are calculated with proposed algorithms [3]. The camera shots can be classified, accordingly to the distance between the camera and the subjects as combinations of close, medium and long shots. The $S = d(s_1, s_2)$ values range between 0.0 (close-up) and 1.0 (long-shot).

Let I set include all the individuals i of the current generation step: $f : I \rightarrow [0, 1]$. Let V_i be the set of an individual's video segments with descriptions, w_d the weight for the specific descriptor d, g the number of genes/segments per individual, and $f_i(V_i)$ the fitness function applied to all those segments:

$$f_i(V_i) = \sum_{v \in V_i} \frac{1}{g}\big(w_C C(v) + w_K K(v) + w_F F(v) + w_S S(v)\big), \qquad i \in I \quad (1)$$

The genetic algorithm flow can be summarized as follows: starts with an initial population where each individual initial fitness value is set, and then the evolutionary loop begins. The validation for the goal achievement is applied, and if any individual is the solution for the problem then the loop ends. If not, a fitness evaluation is used for the selection step, applying a method based on each individual fitness value and the probability of entering a tournament. Several individuals, depending on the selection probability, are elected for mating, and a crossing over technique is applied pairing two individuals. Elitism, if used, guarantees the election of the best individuals. mutation is the next step, and individuals that were mated may be mutated with a very low probability. For the ones that weren't mated, a shortcut towards the step of elimination is taken. Next, a choice of individuals to be eliminated is made and the ones that are chosen are disposed of. The ones that stay are the population new generation.

4 MovieGene System

The MovieGene's system architecture (figure 2) considers three components:

Interface. The user interface is a Java application/applet. It is a window providing interactive access to the evolving videos that are to be produced.

Application. The core is the MovieGene's application linked with libraries for specific tasks: VideoMPEG7 for reading and writing multimedia documents descriptors in MPEG-7 format and coded in an XML file; JMF[2] which provides low level methods for multimedia objects reading, writing and editing; ECJ[3] for the implementation of the genetic algorithms module.

[2] Java Media Framework – `http://java.sun.com/products/java-media/jmf/`

[3] Java-based evolutionary library – `http://cs.gmu.edu/~eclab/projects/ecj/`

Repository. The container for the original multimedia documents and also for the new produced documents along with the respective media descriptors.

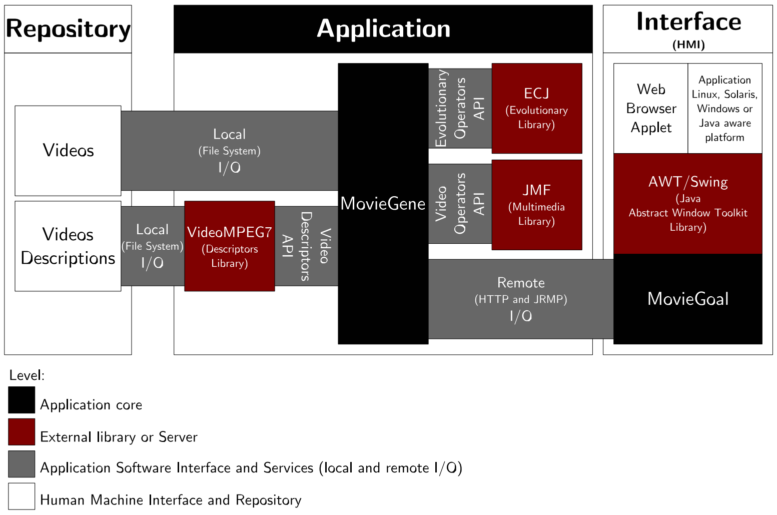

Repository
Application
Interface
(HMI)
Videos
Videos Descriptions
Local (File System) I/O
Local (File System) I/O
VideoMPEG7 (Descriptors Library)
Video Descriptors API
MovieGene
Evolutionary Operators API
ECJ (Evolutionary Library)
Video Operators API
JMF (Multimedia Library)
Remote (HTTP and JRMP) I/O
Web Browser Applet
Application Linux, Solaris, Windows or Java aware platform
AWT/Swing (Java Abstract Window Toolkit Library)
MovieGoal
Level:
Application core
External library or Server
Application Software Interface and Services (local and remote I/O)
Human Machine Interface and Repository

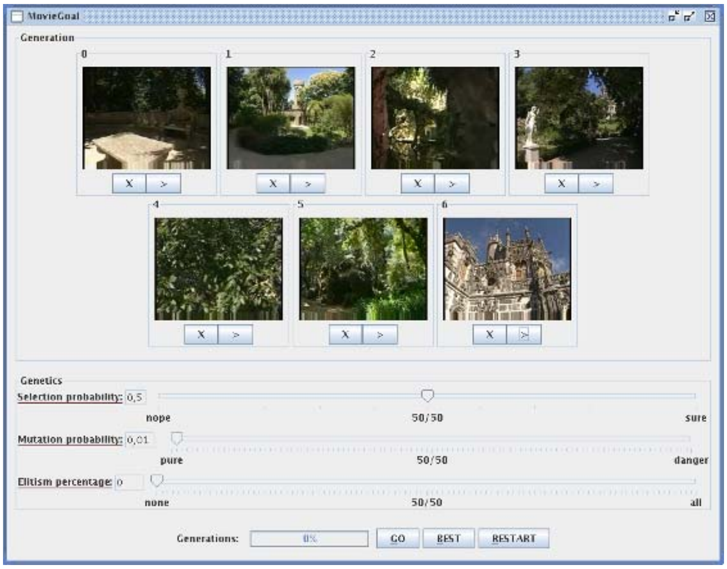
Fig. 2. MovieGene's system architecture

In the initial screen the user can introduce the intended characteristics for the final document, including the semantic description, the shot type, and the color histogram, using text boxes, histogram sliders and buttons. Genetic parameters are the probabilities of selection for crossing over and mutation, the percentage of elitism, and the number of generations. At each step of the evolutionary loop, the resulting videos are presented (figure 3). The user can eliminate a specific video, hit for the next round, or go towards process completion, when the resulting (best) video can be played.

Several unattended tests were done with preset genetic parameters: selection for crossover probability to $p_S = 0.5$, mutation probability to $p_M = 0.01$, elitism to 10%. The selection method used was 2 Tournament and combination was One-Point crossover. The default weights were: $w_C = 0.1$, $w_F = 0.2$, $w_K = 0.6$ and $w_S = 0.1$. The tests showed that the results closely match the goals defined by the user in combining video segments and that the genetic algorithm can be used interactively, taking less than a second to present the results of each generation to the user.

5 Conclusions and Perspectives

A new multimedia authoring paradigm was introduced for the production of video-based documents, using evolutionary computation, video annotation and



Selection probability: 0,5

Mutation probability: 0,01

Elitism percentage: 0

Generations: 0%

Fig. 3. Evolutionary step screen

cinema editing properties. The evolutionary process aims at best clips editing, according to a fitness function based on distance metrics for color, camera and textual annotation descriptors.

Currently, MovieGene allows for the accommodation of spatial, graphical, and rhythmic continuity editing rules, mainly through the support of syntactic properties like color and shot distance, and of additional editing rules that may rely on more semantic properties. Although it may perform automatic video editing, MovieGene intends to empower the user as a film editor, supporting the creative edition by proposing innovative evolutionary combinations the user may subjectively select from, in the process of arriving at more satisfactory or artistic solutions. Some of the improvements in this direction include the provision of a more abstract, rich, and flexible interface, that does not require the user to be aware of low level video descriptors and genetic algorithms terminology.

References

1. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. (1989)
2. Manjunath, B., Salembier, P., Sikora, T.: Introduction to MPEG-7 Multimedia Content Description Interface. John Wiley & Sons, Ltd, West Sussex, EN (2002)
3. Henriques, N.A.C.: MovieGene: A multimedia production system using evolutionary computation. Master's thesis, Faculty of Sciences and Technology of the New University of Lisbon (2005)

Audible Convergence for Optimal Base Melody Extension with Statistical Genre-Specific Interval Distance Evaluation

Ronald Hochreiter

Department of Statistics and Decision Support Systems, University of Vienna,
Universitätsstraße 5/9, Vienna 1010, Austria

Abstract. In this paper, an evolutionary algorithm is used to calculate optimal extensions of a base melody line by statistical interval-distance minimization. Applying an evolutionary algorithm for solving such an optimization problem reveals the effect of audible convergence, when iterations of the optimization process, which represent sub-optimal melody lines, are combined to a musical piece. An example is provided to evaluate the algorithm, and to point out differences, when different musical genres, represented by different interval distance classification schemes, are applied.

1 Introduction

With the progress of computers, various compositional methods were converted to computational algorithms, and composers started to apply more and more (especially stochastic) methods to achieve an automatic generation of music. See [1] for an overview of methods used in the past.

Recently, Operations Research methods have been applied to generate optimal melody lines, e.g. in [2] combinatorial optimization methods were proposed. Evolutionary algorithms are also a valuable approach for the generation of music, see e.g. [3], [4], [5], and the references therein. The main problem with optimization approaches in composition is the definition of the optimum. An intuitive approach goes as follows: the main objectives are specified by the composer, and constraints are implied by some given set of compositional rules. These rules depend on the respective area of composition. The decision process for automatic composition is two-fold. First, a mapping of compositional rules to a numerical model, which is suitable for automatic optimization has to be defined. After the optimization has been conducted, a re-mapping from the numerical optimization result to a musical piece has to be applied.

This paper is organized as follows. Section 2 provides a short review of how horizontal tension of melodies is handled numerically. Section 3 presents details of the evolutionary algorithm, which was developed to optimize melodies. Combining intermediate steps of the optimization process to a musical piece leads to the effect of audible convergence, which is exemplified in Section 4. Section 5 concludes the paper.

2 Algorithmic Treatment of Notes and Tension

Let n be an integer value of a musical note, whose value denotes the number of half steps from the lowest C. The distance $i(n_1, n_2)$ between two notes n_1, n_2 is called interval. In some simplified view, the intervals of a melody line, i.e. the horizontal alignment of notes, may be used to evaluate consonance or dissonance. Intervals can be classified with respect to the musical area, e.g. one Classical and one Jazz classification proposed in [6] are shown in Table 1. The perfect fourth ($i(\cdot,\cdot) = 5$) is often regarded as a special case. In Classical context, it is commonly considered as a complete consonant with the main tone of the key of the respective melody above (notated 5^a), while it could also be understood as dissonant with the main tone below (notated 5^b). i^+ denotes an augmentation and i° a diminishment.

Table 1. Interval categories c^c (Classical) and c^j (Jazz)

c^c	Category	Intervals	c^j	Category	Intervals
1	Complete consonants:	0, 5^a, 7, 12	1	Primary consonants	3, 4, 8, 9
2	Incomplete consonants:	3, 4, 8, 9	2	Secondary consonants	0, 7, 12
3	Dissonants:	1,2, 5^b, 6, 10, 11	3	Mild dissonants	2^+, 2, 6, 8^+, 10
			4	Dissonants	1, 11, 13
			x	Perfect fourth	5

The idea is to map intervals i to ordinally scaled categories c shown in Table 1. Using the Classical map would return $c^c(3) = 2$ for an interval $i = 3$, while the Jazz map would return $c^j(3) = 1$. Subsequently, such a numerical evaluation scheme can be applied for optimizing a melody line constrained to horizontal tension. An excerpt from Wolfgang Amadeus Mozart's *Allegro from Eine Kleine Nachtmusik* is depicted in Figure 1. The intervals of this melody are $\|5^a, 5^a, 5^a|5^a, 5^a, 5^a, 4, 3|(2), 3, 3, 3, 3|3, 3, 3, 3, 7\|$.

Applying the category schemes above yields to the interval category list $\|1, 1, 1|1, 1, 1, 2, 2|3, 2, 2, 2|2, 2, 2, 2, 1\|$ with the Classical evaluation scheme, and to $\|x, x, x|x, x, x, 1, 1|3, 1, 1, 1|1, 1, 1, 1, 1, 2\|$ using the Jazz evaluation scheme. The arithmetic mean and variance (in parenthesis) for each bar is 1(0), 1.6(0.3), 2.3(0.3), 1.8(0.2) as well as $x(x)$, 1(0), 1.5(1), 1.2(0.2). For further application, some simplified scheme has to be used in order to deal with the perfect fourth in an automated framework, such that unevaluated sections, i.e. the first bar evaluated with the Jazz scheme, do not occur.

One issue regarding the split of the melody into single bars is e.g. the consideration of the parenthesized major 2nd, which should not be included for tension calculations as it is not a direct part of the melody, and not acting as a dissonant note. Although the musically trained eye and ear is able to sort this out easily, simple numerical algorithms, without using any type of artificial intelligence, are not able to distinguish such facts. A possibility is to exclude intervals between bar lines. This strategy excludes valid as well as important intervals, but has been used below.

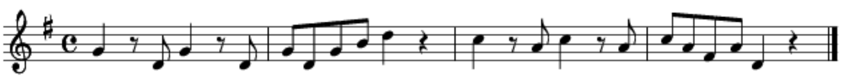


Fig. 1. W.A. Mozart - Allegro from Eine Kleine Nachtmusik (KV525)

3 Evolutionary Algorithms

We use a standard genetic algorithm adapted from [7]. Every member of our population has a numeric representation (genotype) and an audible representation (phenotype). Each chromosome of the population consists of melody data, i.e. one integer per note. To evaluate the fitness of one chromosome, a set of interval categories, as shown above, has to be set up. Furthermore, a target mean and variance structure for each bar is necessary. The target mean and variance values should be in line with the values assigned to each interval category. The fitness f of a generated melody vector is evaluated by calculating weighted sum of the distance between the pre-specified target means μ_i and target variances σ_i^2, i.e.

$$f = \sum_{i=1}^{|b|} \lambda_m \|\mu_i, \text{Mean}(b_i)\| + \lambda_v \|\sigma_i^2, \text{Variance}(b_i)\|$$

where $|b|$ is the number of bars (or bar sets), while λ_m and λ_v can be used to adjust the importance of the mean or the variance. The functions Mean(b_i) and Variance(b_i) return the mean and variance of the interval categories in bar (or bar set) b_i $(i = 1, \dots, |b|)$. Different distances $\| \cdot \|$ can be used.

In [4] ten mutation operators for melody lines are presented, and applied to the MusicBlox project. For the algorithm described in this paper, the following operators are applicable, due to fixed note durations: Swap two adjacent notes, transpose a note pitch by a random interval, transpose a note pitch by an octave, and reverse a group of notes within a randomly selected start and end point.

The algorithm was implemented in MatLab 7. Each generated melody is converted to the GNU LilyPond format, which enables visualization and audibility, as both the score and a MIDI file is generated.

4 Audible Convergence

We aim at constructing an optimization process, which leads to audible convergence, by listening to intermediate steps of the evolutionary algorithm. We start with some random melody, which will be iteratively optimized into a consonant melody, given a subjective set of consonance rules - represented by interval categories. To avoid a pure random noise generation without any musical substance, an optimal extension of some base melody has been used. Assume that some base melody m, which should be extended, consists of n_m notes. Each bar of the final extended melody consists of one note of the base melody and n_e notes of extended melody, such that the final melody has $n = n_m \cdot (n_e + 1)$ notes. These extension

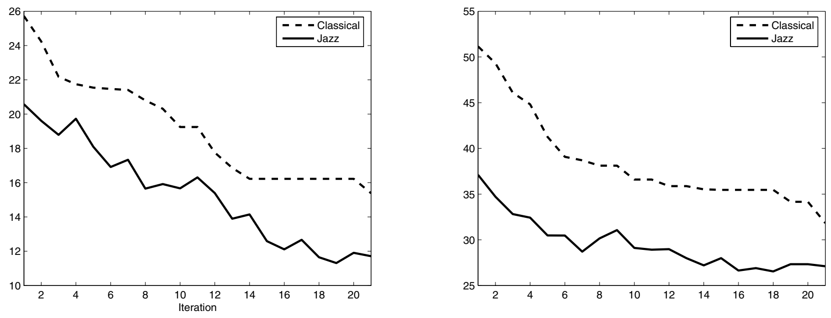


Fig. 2. Example convergence of the algorithm for different weighting schemes

notes will be randomly chosen at the beginning and are meant to converge numerically, and thus also audibly, during iterations of the evolutionary algorithm.

Consider the following example. The first nine notes of Ludwig van Beethoven's Für Elise (Bagatelle in A minor for solo piano (1808), WoO 59) are used as the base melody, i.e. $m = (52, 51, 52, 51, 52, 47, 50, 48, 45)$. Let the melody extension structure in common time be one quarter of the base melody followed by six eights of melody extension. Then the final melody will consist of nine bars with a total of 9×7 notes, i.e. one base note and six extension notes per bar. A simplified interval category scheme was used. The target mean and variance for the first and last three bars is $1(0.5)$, and for the three bars in the middle $3(2)$ has been chosen. This target value structure implies a rather consonant beginning and ending, and a more dissonant middle part.

The initial population has been generated by creating melody lines, where each extension note is modified by adding an uniform-randomly chosen interval in the range of $[-5, 5]$ half steps relative to its base note. The last base note has to be replicated at the end of the score, which is necessary to calculate the fitness of the last bar accordingly.

For evaluating the (negative) fitness, which has to be minimized, the Euclidean distance was chosen. The size of the initial population was set to 30. Within each of 20 iterations, the 10 best of the previous population have been added to the new population. 10 new melodies have been added by mutating five notes of the best melody of the previous population and 10 by mutating ten notes of the second best melody.

The convergence of the fitness value is depicted in Figure 2 for a equally weighted mean and variance ($\lambda_m = \lambda_v = 1$, left), as well as $\lambda_m = 3$ and $\lambda_v = 1$ (right). The audible convergence cannot be shown in this paper satisfactorily, as the size of final scores tends to grow to several pages.

5 Conclusion

In this paper, an evolutionary algorithm to calculate optimal extensions of base melody lines was presented. The algorithm applies a minimization of the sum of distances between pre-defined target means and target variances and the intervals (interval categories) of the melody. Using an evolutionary algorithm reveals the effect of audible convergence, when iterations of the optimization process are combined to a musical piece. Using different interval category schemes for different musical genres results in different audible convergence structures, which lead to musically interesting results. A motivating example was conducted and summarized. This basic algorithm, which contains a set of musical simplifications, can be further refined to obtain even more audibly convergent results. This paper provides the basis for such efforts.

References

1. McAlpine, K., Miranda, E., Hoggar, S.: Making music with algorithms. a case study system. Computer Music Journal **23** (1999) 19–30
2. Schell, D.: Optimality in musical melodies and harmonic progressions: The travelling musician. European Journal of Operations Research **140** (2002) 354–372
3. Gartland-Jones, A.: Can a genetic algorithm think like a composer? 5th International Conference on Generative Art. Milan, Italy. (2002)
4. Gartland-Jones, A.: MusicBlox: A real-time algorithmic composition system incorporating a distributed interactive genetic algorithm. In Cagnoni, S., Romero Cardalda, J., Corne, D., Gottlieb, J., Guillot, A., Hart, E., Johnson, C., Marchiori, E., Meyer, J.A., Middendorf, M., Raidl, G., eds.: Applications of Evolutionary Computing: EvoWorkshops 2003. Volume 2611 of Lecture Notes in Computer Science., Springer (2003) 490–501
5. Manaris, B., Vaughan, D., Wagner, C., Romero, J., Davis, R.: Evolutionary music and the Zipf-Mandelbrot law: Developing fitness functions for pleasant music. In Cagnoni, S., Romero Cardalda, J., Corne, D., Gottlieb, J., Guillot, A., Hart, E., Johnson, C., Marchiori, E., Meyer, J.A., Middendorf, M., Raidl, G., eds.: Applications of Evolutionary Computing: EvoWorkshops 2003. Volume 2611 of Lecture Notes in Computer Science., Springer (2003) 522–534
6. Haunschild, F.: Die Neue Harmonielehre, Teil I. AMA Verlag (1998)
7. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys **35**(3) (2003) 268–308

A Two-Stage Autonomous Evolutionary Music Composer

Yaser Khalifa and Robert Foster

Electrical and Computer Engineering Department,
State University of New York, New Paltz, NY 12561, USA
{khalifay, rfoster}@newpaltz.edu

Abstract. An autonomous music composition tool is developed using Genetic Algorithms. The composition is conducted in two Stages. The first Stage generates and identifies musically sound patterns (motifs). In the second Stage, methods to combine different generated motifs and their transpositions are applied. These combinations are evaluated and as a result, musically fit phrases are generated. Four musical phrases are generated at the end of each program run. The generated music pieces will be translated into Guido Music Notation (GMN) and alternate representation in Musical Instrument Digital Interface (MIDI). The Autonomous Evolutionary Music Composer (AEMC) was able to create *interesting* pieces of music that were both innovative and musically sound.

1 Introduction

In [1], Gartland-Johnes and Colpey distinct between two important objectives of search algorithms; exploration and optimization. Search algorithms, such as Genetic and Evolutionary algorithms, in creative applications definitely serve the formal objective.

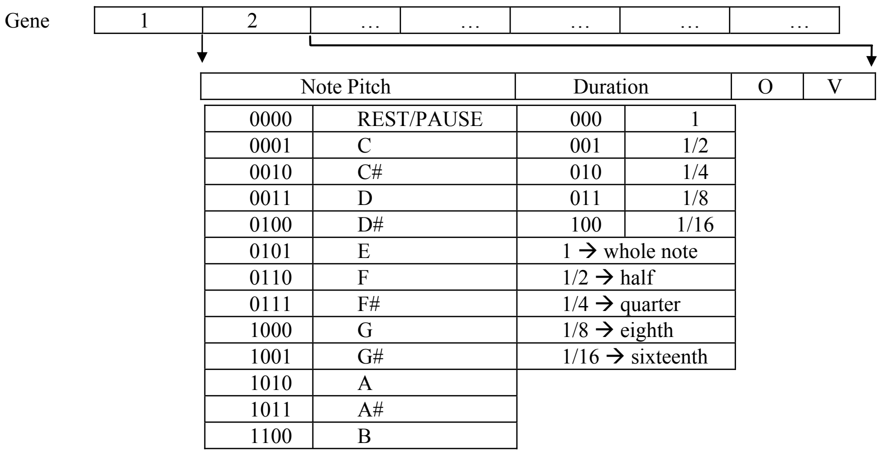
An excellent review of the application of Genetic Algorithms in musical composition is provided in [2]. However, as stated in [3], most evolutionary composition systems listed in literature need a tutor, or a human evaluator in an interactive GA environment. The development of autonomous unsupervised music composers that possess automatic fitness assessment is still limited. Furthermore, the concept of composing music based on a library of motives is, however, near or perhaps slightly beyond the frontier of current capabilities of artificial Intelligence (AI) technology. Thus, this area of research spearheads a new direction in automated composition. For that, designers of evolutionary music requires new techniques that focus on creating classes of musical potential, as opposed to existing techniques that describe predicted linear outcomes. That should lead to an examination of the dynamic interaction between aspects of musical system [4]. The work presented in this paper is an attempt in that direction.

2 Genetic Algorithms Implementation

The composition of music is performed in two Stages. In Stage I, a set of motifs are generated. In Stage II, motifs and their transpositions are combined to form two music phrases A and B. At the end of Stage II, phrase A sharp is generated by sharing each note of the phrase. At the end, a combination of ABA$^{\#}$A is produced, which is one of the common combinations in music composition theory.

3 Stage I

In Stage one, motifs are generated. A table of 16 best motives is constructed to be used in Stage two. These motifs will be used both in their current and transposed locations to generate musical phrases in Stage two. Figure 1, shows the chromosome structure in Stage I. The genetic structure of one gene, of the chromosome used in Stage I, is shown in Figure 1.



Gene	1	2	…	…	…	…	…

Note Pitch	Duration	O	V

0000	REST/PAUSE	000	1
0001	C	001	1/2
0010	C#	010	1/4
0011	D	011	1/8
0100	D#	100	1/16
0101	E	1 → whole note	
0110	F	1/2 → half	
0111	F#	1/4 → quarter	
1000	G	1/8 → eighth	
1001	G#	1/16 → sixteenth	
1010	A		
1011	A#		
1100	B		

Fig. 1. Chromosome and gene structure for Stage I

At the end of Stage I, a table of the top 16 motifs is constructed. Each row in the look-up table represents a motif. The columns represent the different notes in the motif. Although all motifs generated are one whole note in duration, they could be composed of either one, two, four, six, or eight notes.

3.1 Stage I Evaluation Function

In Stage I, only an Intervals Evaluation Function was used. Within a melody line there are acceptable and unacceptable jumps between notes. Any jump between two successive notes can be measured as a positive or negative slope. Certain slopes are acceptable, others are not. The following types of slopes are adopted: *Step*: a difference of 1 or 2 half steps. This is an acceptable transition. *Skip*: a difference of 3 or 4 half steps. This is an acceptable transition. *Acceptable Leap*: a difference of 5, 6, or 7 half steps. This transition must be resolved properly with a third note. *Unacceptable Leap*: a difference greater than 7 half steps. This is unacceptable.

If a leap is acceptable and resolves properly, no penalty will be assigned. There is also a possibility of bonus within the interval section. Certain resolutions between notes are pleasant to hear. We can define these bonus resolutions as the 12-to-13 and the 6-to-5 resolutions. The first is a stronger resolution, and therefore receives a larger weight. Thus the bonuses are calculated as in equations (1) and (2).

$$\text{12-to-13 bonus} = (\text{\#occurances of 12-to-13 steps}/15) * 0.66 \tag{1}$$

$$\text{6-to-5 bonus} = (\text{\#occurances of 6-to-5 steps}/15) * 0.34 \tag{2}$$

The total interval fitness:

$$\text{Interval Fitness} = 1 / (\text{total\_error}\ (1 - \text{total\_bonus})) \tag{3}$$

4 Stage II

In Stage II, motifs from the look-up table constructed in stage I are combined to form two phrases A, and B. Each phrase is eight measures, and each measure is one whole-note motif, Figure 2.

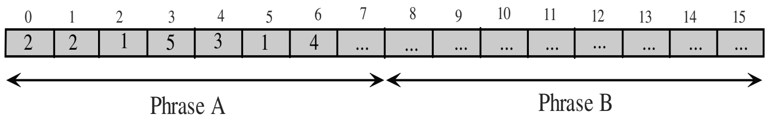


Fig. 2. Chromosome structure for Stage II

4.1 Stage II Evaluation Functions

In stage II, two evaluation functions are implemented; Intervals, and ratio. The Interval evaluation function described in the previous section is used to evaluate interval relationships between connecting notes among motifs. Other evaluation function is described below.

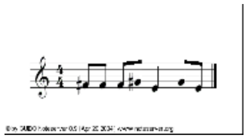
Ratios Evaluation Function. The basic idea for the ratios section of the fitness function is that a good melody contains a specific ideal ratio of notes, and any deviation from that ideal results in a penalty. There are three categories of notes; the tonal centers that make up the chords within a key, the color notes which are the remaining notes within a key, and chromatic notes which are all notes outside a key. Each type of note is given a different weight based on how much a deviation in that portion of the ratio would affect sound quality. The arbitrary ratios sough were: Tonal Centers make up 60% of the melody; Color Notes make up 35% of the melody; and Chromatic Notes make up 5% of the melody. Although these ratios choices could be quite controversial, they were a starting point and current ongoing work is looking further into making these ratios chosen by the user or music style dependent.

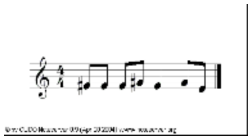
5 Results

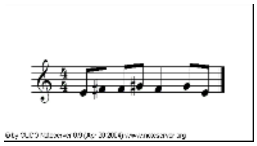
The four motifs in Figure 3 (a) to (d) were picked from the final 16 motifs chosen by the program. It can be observed that each motif has an identical rhythm consisting of four eighth-notes, one quarter-note, and two more eighth notes.

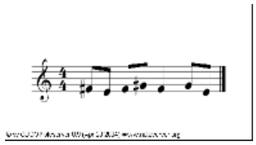
Examining motif *a*, the first three notes are all F$^{\#}$'s, indicating that no penalty will be assigned (a step size of 0). The next note is a G$^{\#}$, (2 half-steps away from F$^{\#}$). This transition is classified as a step and no penalty is assigned. The following notes are F$^{\#}$, G$^{\#}$, and E (a difference of 2, 2, and 3 half steps, respectively). These transitions are also acceptable.

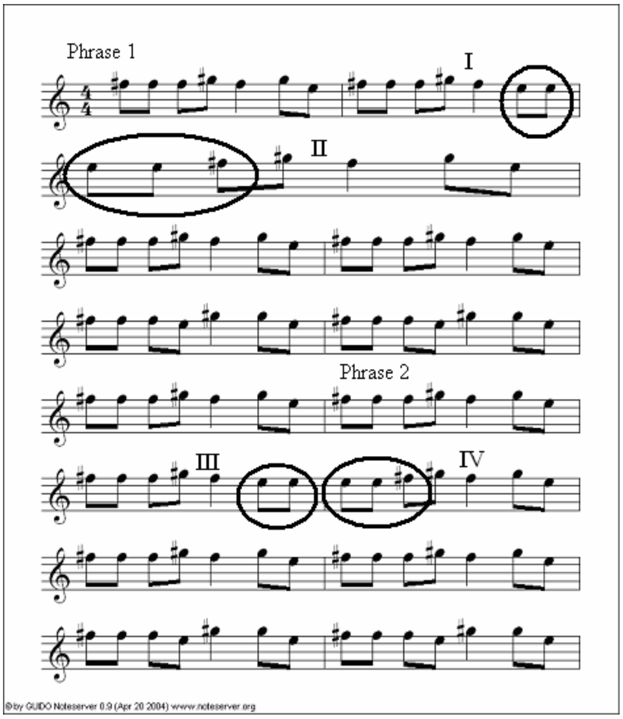
Since each of the motifs in Figure 3 (a) to (d) contained an identical rhythm, it is of no surprise that a piece composed from these motifs contain the same rhythm. What is interesting to note, however, are the measures marked as I, II, III, and IV.


(a)


(b)


(c)


(d)



(e)

Fig. 3. (a) - (d) Sample motifs generated in Stage I of the Evolutionary Music Composer. (e) A partial piece composed from motifs in the same generation as those in (a) through (d).

Measure I and III are the only measures throughout the entire excerpt in which the last two eighth-notes are not G$^{\#}$ and E. Measures II and IV are the only ones in which the first three eighth-notes are not all F$^{\#}$'s. The last note of measure I and the first note of measure II are the same. This is the result of the intervals evaluation function, since it's role in Stage II is to evaluate the transitions between motifs.

6 Discussion and Future Work

New techniques in evaluating combinations of motives are needed. The evaluation of motive combination should take into consideration the overall musical piece rather than the note transition resolutions of the first and last notes in the motif only. One approach that will be investigated is the application of formal grammars. A formal grammar is a collection of either or both descriptive or prescriptive rules for analyzing or generating sequences of symbols. In music, these symbols are musical parameters such as notes and their attributes.

In a multi-objective optimization problem such as music composition, different evaluation functions are applied and contribute in the fitness measure of a generated piece. The main functions that have been designed are intervals and ratios. They have been equally considered in evaluating the evolutionary generated music so far.

References

1. Gartland-Jones, A. Copley, P.: What Aspects of Musical Creativity are Sympathetic to Evolutionary Modeling, Contemporary Music Review Special Issue: Evolutionary Models of Music, Vol. 22, No. 3, (2003), pages 43-55.
2. Burton, A.R. and Vladimirova, T.: Generation of Musical Sequences with Genetic Techniques, Computer Music Journal, Vol. 23, No. 4, (1999), pp 59-73.
3. Miranda, E.R.: At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestra and Origins of Melody, Evolutionary Computation, Vol. 12, No. 2, (2004), pp. 137-158.
4. Miranda, E.R., Composing Music Using Computers, Focal Press, Oxford, UK, (2001).

Layered Genetical Algorithms Evolving into Musical Accompaniment Generation

Ribamar Santarosa[1,2,⋆], Artemis Moroni[1,2], and Jônatas Manzolli[2]

[1] Research Center "Renato Archer",
Campinas, São Paulo Brazil
[2] State University of Campinas — Interdisciplinary Nucleus of
Sound Communication, Campinas, São Paulo Brazil
ribamar@fee.unicamp.br, artemis.moroni@cenpra.gov.br,
jonatas@nics.unicamp.br

Abstract. We present a theoretical evolutionary musical accompaniment generating system capable of evolving to different organized sounds according to an external performer. We present a new approach for implementing the fitness functions.

1 Introduction

Musical accompaniment system design is the task of making a system to produce musical accompaniment for a soloist performer as a human accompanist would produce. The first successful musical accompanist system dates back to 1984 [1]. The musical accompaniment system design can be subdivided into three subtasks: "Listening", "Musical Decision" and "Performing". "Listening" consists in hearing the aural data being played by the soloist and translating it into a notation that the system can interpret. "Musical Decision" is the system's decision of which is its part in the music, id est, what is the musical data the system should play. And "Performing" is the system's part execution itself; commonly referred as *synthesis* for computerized systems. Musical accompaniment system designers, even separating the system in those three subtasks, usually prefer to model the whole system [2]. Others, however, focus on only one of those parts. For example, [3] focuses on "Musical Decision" subtask; our proposal has that same focus.

In order to take ahead the "Musical Decision" part of the system, rather than simply reading a score [1], systems today intend to implement advanced algorithms for musical composition [4], such as use of genetic algorithms (GA).

In despite of criticisms [5], GA continued to be a fluent framework for musical composition [6][7]. [8], another work of the same author group of [5], says *"a GA with no notion of meta-level control of the reasoning process is unlikely to solve the harmonisation problem well"*, however, the works that continued using GA do not seem to care much on that and have used different approaches to attack the eminent problems concerning to the use of GA. [6] used a hybrid approach

⋆ This work was sponsored by CNPQ.

that, somehow, depended on an interactive GA (IGA); [9] and [7] eliminated the fitness function phase.

The problems related to the GAs that lead researchers to use IGAs, or any alternative algorithm, are closely related to the *subjectiveness* of modeling a fitness function, which is, in turn, one of the consequences of the *hardness* of modeling a fitness function. Since a composer changes its set of musical rules even within the same music, the GA may work well untill a certain point , but, with the increase of the music complexity, it becomes hard to trust in a single fitness function.

Here we present a theoretical proposal of an alternative implementation of fitness functions in the direction of counterpart the musicians subjectivity with a layered control of fitness functions, which we will call *"meta fitness function"*. We begin in the Sect. 2 with a overview of the system. In the Sect. 3 we model the basic structures present in the system. In the next step (Sect. 4) we talk about the fitness adaptation layer and introduce the meta fitness function. Then we make a discussion (Sect. 5) and, finally, the conclusion (Sect. 6).

2 Overview

Look at Fig. 1, where we show the parts of the system. The external performer inserts a musical data stream. This stream is passed to the fitness adaptation algorithm, which uses it to modify a fitness function. This modified fitness function is delivered to the genetic algorithm that generates accompaniment data to be played. And, then, the genetic algorithm returns the musical accompaniment to be played.

We will use GA both for generating musical data to be played and to modify the fitness function. We call the former the "Evolutionary Accompaniment Generation" (EAG) layer and the latter the "Fitness Adaptation" (FA) layer.

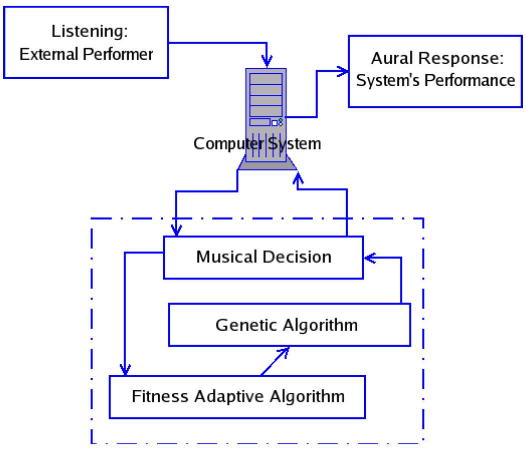


Fig. 1. The parts of the system

For our GAs design, we will let openings for the design of reproduction, crossing-over and mutation methods. So we will only concentrate on the design of the fitness functions.

3 Data Structures Models

Before modeling a fitness function, we must define how the individuals of the GA are. For the EAG layer, our proposal does not depend strongly on such a definition, but we must provide one for consistency reasons. We will use a very simple definition: An individual is a set of pitchs, which we call *"cluster"*.

For the FA layer, each individual must represent a fitness function for the EAG layer, once we want the EAG fitness function to be time-variant. To get this, we state that an individual of the FA layer is a set of rules that guides the fitness function of the EAG layer. This set of rules are filled in two tables:

Rhythmic Pattern Table (RPT). The RPT has information about the rhythm and its relationship with the clusters. In this table, each column represents a rhythmic unit. The first line tells, in an arbitrary time unit, how long each rhythmic unit lasts. The second line tells the beat strength of each rhythmic unit. From the third line on, for each pitch in the cluster there is a corresponding line telling if the pitch is to be played or not at a rhythmic unit. This example:

first line	1 1 1 1
second line	2 $\frac{1}{2}$ 1 $\frac{1}{2}$
third line	1 0 1 0

is a music part in 2/4 metre: 4 rhythmic units with the same duration. The first rhythmic unit is to be played with a strong beat, the third a soft beat, the second and the fourth have very soft beats. The first pitch in the cluster is to be played at the first and the third rhythmic unit.

Transitional Table (TT). This table is filled with data saying how good is to go from one pitch to another. Let us examine this example with, for simplicity, only the pitchs of the traditional western major diatonic scale:

C D E F G A B	Note:	D E F G A B C
.1 .2 .3 .4 .5 .6 .7	C	.7 .6 .5 .4 .3 .2 .1
D E F G A B C	Note:	E F G A B C D
.1 .2 .3 .4 .5 .6 .7	D	.7 .6 .5 .4 .3 .2 .1
...	...	...

So, descending from C to the prior D is 0.2 good in the arbitrary scale used; and ascending from C to the next D is 0.7 good.

Each entry in the table can have, as well, a pointer to another TT, which permits the fitness function to evaluate sequences of pitchs. If the sequence C, D, E is to be evaluated, and the entry $tt(C, Ascending\ D)$ has a pointer to another TT tt', so, the value of $tt'(D, Ascending\ E)$ is considered instead of $tt(D, Asceding\ E)$.

4 The FA Layer: The Meta-fitness Function

Regard again Fig. 1. The FA layer – the Fitness Adaptation Algorithm in the figure – receives a musical stream, which is being externally performed. Basing on this stream, it chooses a new individual representing a fitness function to govern the EAG layer.

The algorithm expects the musical stream to be similar to the musical stream received up to the present moment, and keeps generating and selecting individuals according to the musical stream received up to the present moment. The system must be prepared, however: the environment can suddenly change and there may not be enough time for the evolution to work on the avaliable data. For the case of abrupt changes, the system maintains a database of diversified individuals. The algorithm for the FA layers fitness function, which we call *"meta-fitness function"*, is shown in the Alg. 1.

Algorithm 1. Meta-Fitness Function

```
1: ffi means "individual representing a fitness functions for the EAG layer".
2: Let $P$ be the generated population of ffi.
3: Let $D$ be the database of ffi.
4: Let $\boldsymbol{f}_c$ be the current ffi.
5: $E = D + \{\boldsymbol{f}_c\}$
6: Let $\boldsymbol{m}$ be the musical stream being inserted.
7: Let $\boldsymbol{f}_\alpha$ be such $\boldsymbol{f}_\alpha(\boldsymbol{m}) = max\{\boldsymbol{s}_i(\boldsymbol{m}), \forall \boldsymbol{s}_i \in E\}$
8: if $\boldsymbol{f}_\alpha = \boldsymbol{f}_c$ then
9:     Let $\boldsymbol{f}$ be such $-\|\boldsymbol{f}_c - \boldsymbol{f}\| = max\{-\|\boldsymbol{f}_c - \boldsymbol{f}_i\|, \forall \boldsymbol{f}_i \in P\}$ //smooth change
10: else
11:     $\boldsymbol{f} \leftarrow \boldsymbol{f}_\alpha$ //abrupt change
12: end if
13: return $f$
```

The "distances between two individuals" are euclidian distances between the values in the data structures. [1] The meta-fitness function recognizes the happening of an abrupt change when there is in the database any individual that fits the musical stream better than the current one, and in this case, it returns this better individual; otherwise it returns the individual most similar to the current one.

[1] It must there be enough care about not comparing data to pointers in TTs.

5 Discussions

The problems related to fitness functions modeling for evolutionary composition arise because we cannot count on having a methodologic way to define such functions. Once the generated data must converge according to the fitness function, any mistake shall lead everything to a system's misbehaviour. Once the music subject is very subjective, the designer of the fitness function is likely to make the system act as the designer thinks to be "good". And once the musical genre or rules can change along the time, a fitness function can stop working. These points lead us to the need of a way to make the fitness function to be corrected along the time, and a meta-fitness function seems to be conceptually good to do this work.

6 Conclusion

The task of modeling fitness functions for GAs in evolutionary composition is hard and delegating the judgement of fitness to human evaluation can be a dull work. GA and IGA have to find ways where human beings interacts with the system, with your natural actions, like singing, dancing, or playing any instrument. GA and IGA shall continue to be a good framework for musical composition, if designers can incorporate elaborated criticisms into theirs systems.

The implementation of this work is in progress.

References

1. Dannenberg, R.: An on-line algorithm for real-time accompaniment. Proceedings of the International Computer Music Conference (1984)
2. Raphael, C.: Orchestra in a box: A system for real-time musical accompaniment. IJCAI (2003)
3. Bryson, J.: The reactive accompanist: Adaptation and behavior decomposition in a music system. The Biology and Tech. of Intelligent Autonomous Agents (1994)
4. Papadopoulos, G., G., W.: AI methods for algorithmic composition: A survey, a critical view and future prosp. Symposium on AI and Scientific Creativity (1999)
5. Wiggins, G., Papadopoulos, G., Phon-Amnuaisuk, S., Tuson, A.: Evolutionary methods for musical composition. I. Journal of Comp. Anticipatory Systems (1999)
6. De Felice, F., Abbattista, F., F., S.: Genorchestra: An interactive evolutionary agent for musical composition. Generative Art and Design Conference (2002)
7. Biles, J.: Autonomous GenJam: Eliminating the fitness bottleneck by eliminating fitness. Genetic and Evolutionary Computation Conference (2001)
8. Phon-Amnuaisuk, S., Tuson, A., Wiggins, G.: Evolving musical harmonization. ICANNGA (1999)
9. Moroni, A., Manzolli, J., Von Zuben, F.: Evolution and ARTbitration. Procedings of the International Conference on Computer Graphics and Artificial Inteligence, Limoges, Frana (2000) 141–146

A Preliminary Study on Handling Uncertainty in Indicator-Based Multiobjective Optimization

Matthieu Basseur[1] and Eckart Zitzler[2]

[1] LIFL/CNRS/INRIA, Bat M3 Cité Scientifique,
Villeneuve d'Ascq 59655, France
basseur@lifl.fr

[2] Computer Engineering and Networks Laboratory, ETH Zürich,
35 Gloriastrasse, Zurich 8092, Switzerland
zitzler@tik.ethz.ee.ch

Abstract. Real-world optimization problems are often subject to uncertainties, which can arise regarding stochastic model parameters, objective functions and decision variables. These uncertainties can take different forms in terms of distribution, bound and central tendency.

In the multiobjective context, several studies have been proposed to take uncertainty into account, and most of them propose an extension of Pareto dominance to the stochastic case. In this paper, we pursue a slightly different approach where the optimization goal is defined in terms of a quality indicator, i.e., an objective function on the set of Pareto set approximations. We consider the scenario that each solution is inherently associated with a probability distribution over the objective space, without assuming a 'true' objective vector per solution. We propose different algorithms which optimize the quality indicator, and preliminary simulation results indicate advantages over existing methods such as averaging, especially with many objective functions.

1 Motivation

Knowledge about the set of Pareto-optimal solutions is useful in many applications involving multiple objectives. Therefore, considerable research, particularly in the context of evolutionary computation, has been devoted to generating methods, i.e., techniques that try to generate the entire Pareto set or approximations of it. One recent approach of this type is indicator-based multiobjective optimization [1], which has the advantage that no additional diversity preservation mechanisms are required. Zitzler and Künzli [1] have demonstrated that this approach can be superior to popular algorithms such as SPEA2 and NSGA-II, with respect to the quality indicator under consideration.

Many real-world optimization problems are subject to uncertainties and therefore this aspect needs to be accounted for. Among the different types of uncertainty one can distinguish, cf. [2], we here consider the case that the determination of the objective function values is a stochastic process, i.e., every time a solution is evaluated, a different objective vector may be returned. Such a scenario emerges, e.g., if the underlying computational model involves stochastic components such as Monte Carlo simulation.

While uncertainty in the objective functions gained some attention in the single-objective context [3, 2], only few studies address this problem within a multiple criteria setting. [4] were among the first to discuss uncertainty in the light of generating methods, although they did not propose a particular multiobjective optimizer for this purpose. Several years later, [5] and [6] independently proposed stochastic extensions of Pareto dominance and suggested similar ways to integrate probabilistic dominance in the fitness assignment procedure; both studies consider special types of probability distributions. In [7], another ranking method is proposed which is based on the average value per objective and the variance of the set of evaluations. Similarly, [8] suggested to consider for each dimension the mean over a given sample of objective vectors and to apply standard multiobjective optimizers for deterministic objective functions.

In this paper, we consider different scenarios for uncertain environments and propose and investigate several techniques to integrate uncertainties within the framework of indicator-based search, based on the algorithm presented in [1]; here, we focus on the ϵ_+ quality indicator [9]. In particular, the main contributions are:

- A probabilistic model that combines quality indicators and uncertainty;
- Different algorithms to integrate this model into the optimization process;
- Preliminaries experimentations on multiobjective test functions to compare these algorithms to existing ones.

The major differences to previous studies are (i) the investigation of uncertainty in the context of indicator-based multiobjective search and (ii) the more general perspective, as knowledge about the type of underlying probability distribution is not required.

2 Proposed Model: Combining Uncertainty and Binary Indicators

2.1 Indicator-Based Multiobjective Optimization

Let X denote the search space of the optimization problem under consideration and Z the corresponding objective space. Without loss of generality, we assume that $Z = \mathbb{R}^n$ and that all n objectives are to be minimized. In the absence of uncertainty, each $\mathbf{x} \in X$ is assigned exactly one objective vector $\mathbf{z} \in Z$ on the basis of a vector function $f : X \to Z$ with $\mathbf{z} = f(\mathbf{x})$. The mapping f defines the evaluation of a solution $\mathbf{x} \in X$, and often one is interested in those solutions that are Pareto optimal with respect to f.[1] However, generating the entire set of Pareto-optimal solutions is usually infeasible, e.g., due to the complexity of the underlying problem or the large number of optima. Therefore

[1] A solution $\mathbf{x} \in X$ is Pareto optimal if and only if there exists no $\mathbf{x}' \in X$ such that (i) $f(\mathbf{x}')$ is component-wise smaller than or equal to $f(\mathbf{x})$ and (ii) $f(\mathbf{x}') \neq f(\mathbf{x})$.

in many applications, the overall goal is to identify a good approximation of the Pareto-optimal set.

Different notions of what a good Pareto set approximation is are possible, and the definition of approximation quality strongly depends on the decision maker and the optimization scenario. We here assume that the optimization goal is given in terms of a binary quality indicator I, as proposed in [1]. A binary quality indicator, cf. [9], is a function $I : \mathcal{M}(Z) \times \mathcal{M}(Z) \to \mathbb{R}$, where $\mathcal{M}(Y)$ stands for the set of all possible multisets over Y, that can be regarded as a continuous extension of the concept of Pareto dominance to multisets of objective vectors. The value $I(A, B)$ quantifies the difference in quality between $A, B \in \mathcal{M}(Z)$. Now, if R denotes the set of Pareto-optimal solutions (or any other reference set), and $f(Y) := \{f(\mathbf{x}) \,|\, \mathbf{x} \in Y\}$, then the overall optimization goal can be defined as

$$\mathrm{argmin}_{S \in \mathcal{M}(X)}\ I(f(S), f(R)) \tag{1}$$

2.2 Handling Uncertainty

In the following, the above optimization model will be extended to take uncertainty into account; later, we will discuss how to estimate and compute expected indicator values for uncertain environments.

As to uncertainty, the basic difference to the classical settings is that the vector function f does not represent a deterministic mapping from X to Z, but a stochastic process: every time a solution $\mathbf{x} \in X$ is evaluated using f, it may be mapped to a different objective vector $\mathbf{z} \in Z$. The higher the degree of uncertainty, the larger the variance among the objective vectors resulting from multiple, independent evaluations of $\mathbf{x}$. Thus, with each solution $\mathbf{x}$ a random variable $\mathcal{F}(\mathbf{x})$ is associated the range of which is Z; the underlying probability density function is usually unknown and may be different for other solutions.

Now, consider an arbitrary solution multiset $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \in \mathcal{M}(X)$. Based on the random variables $\mathcal{F}(\mathbf{x}_i)$ associated with the elements $\mathbf{x}_i$ of S, a corresponding random variable $\mathcal{F}(S)$ is defined for S which takes values in $\mathcal{M}(Z)$; $P(\mathcal{F}(S) = A)$ denotes the probability that (i) all members of S are mapped to elements of $A \in \mathcal{M}(Z)$ and (ii) there is at least one $\mathbf{x} \in S$ per $\mathbf{z} \in A$ for which $\mathbf{z} = f(\mathbf{x})$. Using this notation, we can now reformulate the optimization goal for uncertain environments as

$$\mathrm{argmin}_{S \in \mathcal{M}(X)}\ E(I(\mathcal{F}(S), \mathcal{F}(R))) \tag{2}$$

where R is an arbitrary reference set from $\mathcal{M}(X)$ and $E(\cdot)$ stands for the expected value.

Note that there is a fundamental difference to other approaches, cf. [2]: we do not assume that there is a 'true' objective vector per solution which is blurred by noise; instead, we consider the scenario that each solution is inherently associated with a probability distribution over the objective space.

2.3 Estimating the Expected Indicator Value

If the probability density functions are known in advance and identical for all solutions $\mathbf{x} \in X$, then the expected value for any indicator can be computed according to

$$\begin{aligned} E(I(\mathcal{F}(S), \mathcal{F}(R))) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{pdf}_{\mathcal{F}(S)\mathcal{F}(R)}(A, B) \cdot I(A, B) \ \mathrm{d}A\mathrm{d}B \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{pdf}_{\mathcal{F}(S)}(A) \cdot \mathrm{pdf}_{\mathcal{F}(R)}(B) \cdot I(A, B) \ \mathrm{d}A\mathrm{d}B \end{aligned} \tag{3}$$

since $\mathcal{F}(S)$ and $\mathcal{F}(R)$ are independent from each other. Here, $\mathrm{pdf}_{\mathcal{F}(\cdot)}$ denotes the probability density function associated with the random variable $\mathcal{F}(\cdot)$.

However, in practice the underlying probability density functions are in general unknown, may vary for different solutions, and therefore can only be estimated by drawing samples. Let us assume that $\mathcal{S}(\mathbf{x}) \in \mathcal{M}(Z)$ represents a finite sample, i.e., a multiset of objective vectors, for solution $\mathbf{x}$. Now, the expected indicator value $E(I(\mathcal{F}(\mathbf{x}), \{A^*\})$ of $\mathcal{F}(\mathbf{x})$ with respect to a given set of objective vectors $\{\mathbf{z}_1^*, \ldots, \mathbf{z}_q^*\}$ can be estimated as follows

$$\hat{E}(I(\mathcal{F}(\mathbf{x}), \{\mathbf{z}_1^*, \ldots, \mathbf{z}_q^*\})) = \sum_{\mathbf{z} \in \mathcal{S}(\mathbf{x})} \frac{I(\{\mathbf{z}\}, \{\mathbf{z}_1^*, \ldots, \mathbf{z}_q^*\})}{|\mathcal{S}(\mathbf{x})|} \tag{4}$$

where $\hat{E}$ stands for the estimated expected value and $|\cdot|$ for the cardinality of a set. For a multiset S of solutions with $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, the formula is

$$\hat{E}(I(\mathcal{F}(S), \{\mathbf{z}_1^*, \ldots, \mathbf{z}_q^*\}) = \sum_{\mathbf{z}_1 \in \mathcal{S}(\mathbf{x}_1)} \cdots \sum_{\mathbf{z}_m \in \mathcal{S}(\mathbf{x}_m)} \frac{I(\{\mathbf{z}_1, \ldots, \mathbf{z}_m\}, \{\mathbf{z}_1^*, \ldots, \mathbf{z}_q^*\})}{\prod_{1 \leq i \leq m} |\mathcal{S}(\mathbf{x}_i)|} \tag{5}$$

and if one considers a reference set R of solutions with $R = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_r^*\}$, then the estimate amounts to

$$\hat{E}(I(\mathcal{F}(S), \mathcal{F}(R)) = \sum_{\mathbf{z}_1^* \in \mathcal{S}(\mathbf{x}_1^*)} \cdots \sum_{\mathbf{z}_r^* \in \mathcal{S}(\mathbf{x}_r^*)} \frac{\hat{E}(I(\mathcal{F}(S), \{\mathbf{z}_1^*, \ldots, \mathbf{z}_r^*\}))}{\prod_{1 \leq i \leq r} |\mathcal{S}(\mathbf{x}_i^*)|} \tag{6}$$

This approach is based on the assumption that the probability of a solution $\mathbf{x} \in X$ to be mapped to any objective vector $\mathbf{z}$ in the corresponding sample $\mathcal{S}(\mathbf{x})$ is uniformly distributed, i.e., $P(\mathcal{F}(\mathbf{x}) = \{\mathbf{z}\}) = 1/|\mathcal{S}(\mathbf{x})|$ for all $\mathbf{z} \in \mathcal{S}(\mathbf{x})$.

2.4 Computing Expected Indicator Values

Computing the expected indicator value for two multisets of solutions in the aforementioned manner is usually infeasible due to combinatorial explosion. Suppose each multiset contains 100 solutions with a sample size of 10 each, then equation 6 contains $100^{10} \cdot 100^{10} = 10^{40}$ summands. However, if particular properties of the indicator used can be exploited, then the exact calculation for $\hat{E}(\ldots)$ may become feasible. We here propose an algorithm for the (additive) ϵ-indicator [9] to compute the expected quality difference between a multiset $S \in \mathcal{M}(X)$

and a reference set R with one element only - for reference sets of arbitrary size the computation is still too expensive to be useful in practice. Later in Section 3 it will be discussed how this procedure can be integrated into an evolutionary algorithm.

For a minimization problem, the ϵ-indicator $I_{\epsilon+}$ is defined as follows:

$$\begin{aligned} I_{\epsilon+}(A,B) = \inf\nolimits_{\epsilon \in \mathbb{R}}\{\, & \forall \mathbf{z}_2 = (z_{2_1}, \ldots, z_{2_n}) \in B\ \exists \mathbf{z}_1 = (z_{1_1}, \ldots, z_{1_n}) \in A\ : \\ & \forall 1 \le i \le n\ :\ z_{1_i} \le \epsilon + z_{2_i}\} \end{aligned} \tag{7}$$

It gives the minimum ϵ-value by which B can be moved in the objective space such that A is at least as good as B; a negative value implies that A is better than B in the Pareto sense. If B consists of a single objective vector $\mathbf{z}^*$, then the formula reduces to

$$I_{\epsilon+}(A, \{\mathbf{z}^*\}) = \inf_{\epsilon \in \mathbb{R}}\{\exists \mathbf{z}_1 = (z_{1_1}, \ldots, z_{1_n}) \in A\ :\ \forall 1 \le i \le n\ :\ z_{1_i} \le \epsilon + z^*_i\} \tag{8}$$

Now, to compute $\hat{E}(I_{\epsilon+}(\mathcal{F}(S), \{\mathbf{z}^*\}))$ it is not necessary to consider all combinations of objective vectors to which the elements $\mathbf{x} \in S$ could be mapped to. Instead, one can exploit the fact that always the minimum $I_{\epsilon+}(\{\mathbf{x}\}, \{\mathbf{z}^*\}))$-value determines the actual indicator value. By sorting the objective vectors beforehand, it suffices to consider the ϵ-values in increasing order.

In detail, this works as follows. We consider all pairs $(\mathbf{x}_j, \mathbf{z}_k)$, where $\mathbf{x}_j \in S$ and $\mathbf{z}_k \in \mathcal{S}(\mathbf{x}_j)$, and sort them in increasing order regarding the indicator value $I_{\epsilon+}(\{\mathbf{z}_k\}\{\mathbf{z}^*\})$. Suppose the resulting order is $(\mathbf{x}_{j_1}, \mathbf{z}_{k_1}), (\mathbf{x}_{j_2}, \mathbf{z}_{k_2}), \ldots, (\mathbf{x}_{j_l}, \mathbf{z}_{k_l})$. Then, the estimate of the expected indicator value is

$$\begin{aligned} \hat{E}(I_{\epsilon+}(\mathcal{F}(S), \{\mathbf{z}^*\})) = {} & I_{\epsilon+}(\{\mathbf{z}_{k_1}\}, \{\mathbf{z}^*\}) \cdot P(\mathcal{F}(\mathbf{x}_{j_1}) = \mathbf{z}_{k_1}) + \\ & I_{\epsilon+}(\{\mathbf{z}_{k_2}\}, \{\mathbf{z}^*\}) \cdot P(\mathcal{F}(\mathbf{x}_{j_2}) = \mathbf{z}_{k_2} \,|\, \mathcal{F}(\mathbf{x}_{j_1}) \ne \mathbf{z}_{k_1}) + \\ & \ldots \\ & I_{\epsilon+}(\{\mathbf{z}_{k_l}\}, \{\mathbf{z}^*\}) \cdot P(\mathcal{F}(\mathbf{x}_{j_l}) = \mathbf{z}_{k_l} \,|\, \forall_{1 \le i < l} \mathcal{F}(\mathbf{x}_{j_i}) \ne \mathbf{z}_{k_i}) \end{aligned}$$

It can be done more efficiently as soon as if all $I_{\epsilon+}$ values of the different elements of one solution are smaller than $\overline{\epsilon}$, then the remaining $I_{\epsilon+}$ values greater than $\overline{\epsilon}$, have a probability of 0. This scheme can be used, as detailed by Alg. 1.

Algorithm 1 (Estimation of the Expected ϵ-Indicator Value)

```
Input:   $S \in \mathcal{M}(X)$                           (multiset of decision vectors)
         $\mathbf{z}^* \in Z$                               (reference objective vector)
Output:  $\hat{E}(I_{\epsilon+}(\mathcal{F}(S), \{\mathbf{z}^*\}))$   (estimate for the expectation value of $I_{\epsilon+}$)

Step 1:  Determine $\overline{\epsilon} = \min_{\mathbf{x} \in S} \max_{\mathbf{z} \in \mathcal{S}(\mathbf{x})} I_{\epsilon+}(\{\mathbf{z}\}, \{\mathbf{z}^*\})$
Step 2:  Set $L = \emptyset$. For each $\mathbf{x} \in S$ and $z \in \mathcal{S}(x)$ do:
           1. $\epsilon = I_{\epsilon+}(\{\mathbf{z}\}, \{\mathbf{z}^*\})$.
           2. If $\epsilon \le \overline{\epsilon}$ then $\mathsf{AppendToList}(L, (\epsilon, \mathbf{x}))$.
Step 3:  Sort $L$ in increasing order according to the $\epsilon$-values.
Step 4:  Set $\hat{E} = 0$. For each $\mathbf{x} \in X$ set $N[\mathbf{x}] = 0$.
Step 5:  While $\mathsf{NotEmpty}(L)$ do:
           1. $(\epsilon', \mathbf{x}') = \mathsf{FirstElement}(L)$.
           2. $p = 1/(|\mathcal{S}(\mathbf{x}')| - N[\mathbf{x}']) \cdot \prod_{\mathbf{x} \in S} 1 - N[\mathbf{x}]/|\mathcal{S}(\mathbf{x})|$.
           3. $\hat{E} = \hat{E} + p \cdot \epsilon'$.
           4. $N[\mathbf{x}'] = N[\mathbf{x}'] + 1$ and $\mathsf{RemoveFirstElement}(L)$.
Step 6:  Return $\hat{E}$.
```

3 Algorithm Design

In this section, we discuss on how to integrate algorithm 1 in multiobjective EAs in order to achieve the optimization goal defined in equation 2. The following discussion is based on [1].

In the general case, during the selection process of EAs, we are interested in the case that M solutions need to be removed from the current population, with the goal of maximizing the quality of the remaining solutions, in our case according to the binary indicator $I_{\epsilon+}$. But, clearly, this problem is NP-hard. Therefore, consider mainly the steady-state version ($M = 1$), which corresponds to an evolution strategy $ES(N + 1)$. According to a performance indicator, removing one individual can be solved optimally by deleting the solution $\mathbf{x}_w \in S$ which has the worst EIV value.

With the algorithm 1, the Estimation of the Expected ϵ-Indicator Value (EIV) is computed. Then, we are able to evaluate the quality of different populations against a reference set. As shown in equation 2.4, the algorithm can be used to evaluate the quality of a single solution against a reference set. The quality of a solution $\mathbf{x}_i \in S$ is measured by estimating EIV of $S \setminus \{\mathbf{x}_i\}$ against S, which corresponds to the lost of quality of S if we remove the solution $\mathbf{x}_i$. The general EA is detailed in algorithm 2.

Algorithm 2 (Steady-state $IBEA$ algorithm: EIV)

```
Input:   $N$        (population size)
         $G$        (maximum number of generations)
Output:  $S$        (approximation set)

Step 1:  Initialization: Generate an initial population $S$ of size $N$; set the generation
         counter $g$ to 0.
Step 2:  Fitness assignment: Calculate fitness values of individuals in $S$, i.e., for all
         $\mathbf{x}_i \in S$ set $Fit(\mathbf{x}_i) = \hat{E}(I(\mathcal{F}(S), \mathcal{F}(\{\mathbf{x}_i\}))) = \frac{1}{|\mathcal{S}(\mathbf{x}_i)|} \sum_{\mathbf{z}_i \in \mathcal{S}(\mathbf{x}_i)} \hat{E}(I(\mathcal{F}(S), \{\mathbf{z}_i\}))$
Step 3:  Environmental selection: Remove the individual $\mathbf{x}_w \in S$ with the smallest
         fitness value, i.e., $Fit(\mathbf{x}_w) \leq Fit(\mathbf{x})$ for all $\mathbf{x} \in S$.
Step 4:  Termination: If $g \geq G$ then return $S$.
Step 5:  Mating selection: Perform binary tournament selection on $S$.
Step 6:  Variation: Apply recombination and mutation operators and insert the gen-
         erated individual into the population $S$. Increment the generation counter
         ($g = g + 1$) and go to the Step 2.
```

Let us consider s evaluations for all $x^i \in S$. In order to realize this comparison, the epsilon values of each individual have to be sorted (see algorithm 1). Then the selection process, according to the *EIV* fitness assignment algorithm, has a complexity of $\theta(n(Ns)^2 log(Ns))$ (Ns solutions - Ns elements to sort for every solution to evaluate). In order to reduce this complexity, we develop two methods which approximate the ranking obtained with *EIV* algorithm.

The first approach consists in approximating the sorting step of *EIV* by a Bucket sort (*BCK*). c different values are first defined, with an uniform repartition on the definition interval $[I_{min}, I_{max}]$ of the performance indicator (we could also use the minimum and maximum values computed for the indicator). Then, during the Step 1 of algorithm 1, we compute an approximated indicator value $I_{\epsilon^+}^{BCK}$, to speed up the sorting step by the use of bucket sort. Let c_{range} be defined as $(I_{max} - I_{min})/c$ and $cell = (int)[\frac{(I_{\epsilon^+}(z,z^*) - I_{min})}{c_{range}}]$.

$$I_{\epsilon^+}^{BCK}(z, z^*) = I_{min} + cell * c_{range} \tag{9}$$

The maximum error of this approach is equal to $(I_{max} - I_{min})/c$. The algorithm complexity is in $\theta(nN^2 s(s + c))$.

The second approach consists in approximating the minimum value computed by *EIV* with an exponential function (*Exp*) applied on the different computed indicators values, as realized in [1], without uncertainty:

$$Fit(\mathbf{x}_1) = \sum_{\mathbf{z}_1 \in \mathcal{S}(\mathbf{x}_1)} \sum_{\mathbf{x}_2 \in S \setminus \{\mathbf{x}_1\}} \sum_{\mathbf{z}_2 \in \mathcal{S}(\mathbf{x}_2)} -e^{-I_{\epsilon^+}(\mathbf{z}_2, \mathbf{z}_1)/\kappa} \tag{10}$$

With one evaluation per solution, when *kappa* is close to 0, the corresponding ranking tends to be exactly a lexicographic sorting comparison between all computed indicator values. With several evaluations per solution, the probability of occurrence of each possible indicator is not considered here, but the computational complexity of the algorithm 2 is reduced to $\theta(n(Ns)^2)$.

To evaluate the different scheme proposed, we also propose two alternative algorithms. The first approach envisaged consists in approximate *EIV* fitness assignment function is the Averaging method (*Avg*). First, the average value is computed for each objective function, then the exact algorithm can be easily applied with $|\mathcal{S}(x)| = 1$, $\forall \mathbf{x} \in S$. In fact, in this case, we have the relation $\hat{E}(I(\mathcal{F}(S), \mathcal{F}(\{\mathbf{x}^*\})) = \hat{E}(I(\mathcal{F}(S), \{\mathbf{z}^*\})$, and:

$$Fit(\mathbf{x}^*) = \hat{E}(I_{\epsilon^+}(\mathcal{F}(S), \{\mathbf{z}^*\}) = min\{\mathbf{z} \in \mathcal{S}(\mathbf{x}) \ I_{\epsilon^+}\{\mathbf{z}\}\{\mathbf{z}^*\}\} \tag{11}$$

Then, the algorithm complexity is in $\theta(Ns + N^2)$ (averaging step + indicator values computation).

We also implement the fitness assignment method proposed by Hughes [5], based on the Probabilistic Dominance Relation (*PDR*) between solutions:

$$Fit(\mathbf{x}^*) = \frac{1}{|\mathcal{S}(\mathbf{x}^*)|} * \sum_{\mathbf{z}^* \in \mathcal{S}(x^*)} \sum_{i=1}^{n} \sum_{x \in S \setminus x^*} \frac{1}{|\mathcal{S}(x)|} * \sum_{\mathbf{z} \in \mathcal{S}(x)} inf(\mathbf{z}_i, \mathbf{z}_i^*) \tag{12}$$

with $inf(\mathbf{z}_i, \mathbf{z}_i^*)$ equal to 0 (resp. 0.5, 1) if the i^{th} objective value of $\mathbf{z}$ is smaller (resp. equal, greater) than the i^{th} objective value of $\mathbf{z}^*$. The complexity of PDR fitness assignment algorithm is in $\theta(n * (Ns)^2)$.

For all the different schemes proposed in this section, with use algorithm 2. The fitness assignment step is replaced by the corresponding method. Then, for mating selection, we make a binary tournament between the solutions of S, without the deleted solution $\mathbf{x}_w$. To achieve the tournament step, we compare the solutions according to their fitness value, computed for the selection (which is not exactly the true fitness value, since one solution has been removed from the population).

4 Simulation Results

In the following, we investigate two questions concerning performance of the 5 different algorithms. First, we evaluate these algorithms for one selection step, and compare the loss of quality obtained by each method. secondly, comparison is done on entire runs on multiobjective tests functions.

We present only preliminary results. The uncertainty is defined with known bound, distribution and central tendency. Moreover, performance evaluation are realized by using the true objective vectors value of the output solutions. We would like to make evaluation based expected values as in equation 3, but it is really not feasible.

4.1 Environmental Selection

The test are done with the different approximative and exact selection methods previously described: EIV, BCK, Exp, Avg, PDR.

We evaluate the selection process on randomly generated Pareto population: 100 individuals are generated, with random value for each objective function. Then, we scale the values of each individual $\mathbf{x}$ to obtain $\sum_{i=1}^{n} f_i(\mathbf{x}) = n/2$. In the biobjective case, it corresponds to solutions on the diagonal $[\{1,0\},\{0,1\}]$. Then, for each solution, we generate s different evaluations, by adding a random value (in an interval $[-\sigma, \sigma]$), for each objective vector. For each test, we generate 100 random populations, with 10 evaluations per solution, a uniform noise defined on the interval $[-0.05, 0.05]$, and two objective values. We evaluate the different methods by varying the number of objective functions, the sample size, or the level of uncertainty. The bucket sort was tested with $c = 50$.

Then, we first evaluate the selection process with the exact I_{ϵ^+} value computation, which determines the worst solution $\mathbf{x}_w$. Then, for each approximative fitness assignment algorithm i, we compute the worst solution $\mathbf{x}_{w_i}$. To evaluate the effectiveness of the approximation, we compute the difference, in terms of performance indicator, between the exact and the approximative approach: $I_{\epsilon^+}(S \setminus \{\mathbf{x}_{w_i}\}, S) - I_{\epsilon^+}(S \setminus \{\mathbf{x}_w\}, S)$ (smaller values are better).

The figures 1, 2 and 3 give the results obtained with the different selection methods (BCK, Exp, PDR and Avg) for 4 different number of objective functions, 3 different sample size, and 3 levels of uncertainty. The smaller values are

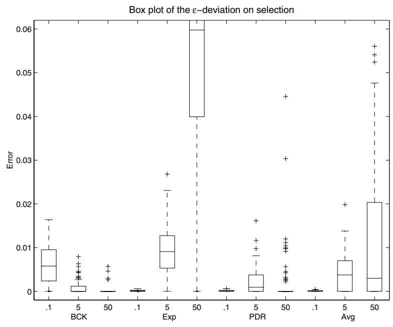


Fig. 1. Average selection error, with different levels of uncertainty

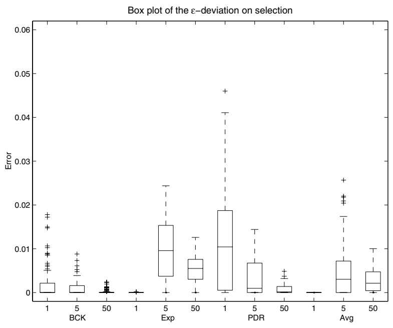


Fig. 2. Average selection error, with different number of evaluations

achieved by BCK in many cases, especially with an important level of uncertainty, many objectives, or a lot of evaluations. The other methods obtain small values only for small sample size or small level of uncertainty. The exponential approaches almost obtain the exact approach results, only when $s = 1$. In normal case, we suggest to use the exponential function approach, which is not expensive to compute and almost gives the optimal results. With uncertainty, BCK seems to be more effective.

4.2 Entire Optimization Runs

For the entire optimization runs, we consider 5 multi-objective test functions taken from the literature: ZDT1, ZDT6 [10], DTLZ2 [11], KUR [12] and COMET [13]. The number of decision variables has been fixed to 50 for all the test problems. Tests are realized by considering two different types of uncertainty: (1) an uniform-distributed random noise is applied on the evaluation functions, in a fixed interval $[-\sigma, \sigma]$; (2) a uniform-distributed random noise is applied on the

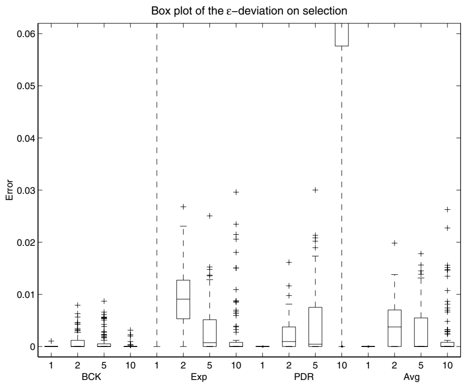


Fig. 3. Average selection error, with different number of objective functions

decision variables. The result is a variable noise, depending on the form of the objective space around the envisaged solution.

The population size N is set to 50, with $s = 5$ evaluations for each solution. Uniform repartition is applied for the two types of uncertainty (i.e. on decision or on objective space). The maximum number of generations is set to 5000. We perform 30 runs for each problem. The different methods are tested with the same initial populations. The other parameters used, such as mutation and recombination operators are those used in [1].

To evaluate the effectiveness of each method, we generate the 'true' objective vector for each solution. Then, for each approximation A, we compute $\hat{E}(I_{\epsilon^+}(R, A))$ value, where R is the reference set, determined by merging all solutions found during the experimentations, and keeping only the non-dominated evaluations. The comparison of the whole set of runs is realized using the Mann-

Table 1. Comparison of the different selection methods for the I_{ϵ^+}-indicator using the Mann-Whitley statistical test: P value, with noise on objective vectors (Z) and on decision vectors (X) - 2 objective problems. A cell 10^{-a} corresponds to a significance level in the interval $[10^{-a}, 10^{-(a-1)}]$.

		EIV		**BCK**		**Exp**		**Avg**		**PDR**	
		Z	X	Z	X	Z	X	Z	X	Z	X
DTLZ2	**EIV**			$> 5\%$	$> 5\%$	10^{-11}	10^{-11}	$> 5\%$	10^{-6}	$> 5\%$	$> 5\%$
	BCK	10^{-4}	$> 5\%$			10^{-11}	10^{-11}	10^{-3}	10^{-3}	$> 5\%$	$> 5\%$
	Exp	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$			$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$
	Avg	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	10^{-11}	10^{-10}			$> 5\%$	$> 5\%$
	PDR	10^{-9}	10^{-4}	10^{-5}	10^{-4}	10^{-11}	10^{-11}	10^{-8}	10^{-8}		
ZDT1	**EIV**			$> 5\%$	$> 5\%$	10^{-4}	10^{-7}	$> 5\%$	$> 5\%$	10^{-5}	$> 5\%$
	BCK	$> 5\%$	$> 5\%$			10^{-5}	10^{-8}	$> 5\%$	$> 5\%$	10^{-7}	$> 5\%$
	Exp	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$			$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$
	Avg	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	10^{-5}	10^{-7}			10^{-6}	$> 5\%$
	PDR	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	10^{-11}	$> 5\%$	$> 5\%$		
ZDT6	**EIV**			$> 5\%$	$> 5\%$	10^{-11}	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$
	BCK	$> 5\%$	$> 5\%$			10^{-11}	$> 5\%$	$> 5\%$	10^{-3}	$> 5\%$	$> 5\%$
	Exp	$> 5\%$	10^{-5}	$> 5\%$	10^{-3}			$> 5\%$	10^{-7}	$> 5\%$	10^{-2}
	Avg	10^{-2}	$> 5\%$	10^{-4}	$> 5\%$	10^{-11}	$> 5\%$			$> 5\%$	$> 5\%$
	PDR	10^{-3}	$> 5\%$	10^{-5}	$> 5\%$	10^{-11}	$> 5\%$	$> 5\%$	10^{-3}		
KUR	**EIV**			$> 5\%$	$> 5\%$	$> 5\%$	10^{-4}	$> 5\%$	10^{-4}	$> 5\%$	$> 5\%$
	BCK	$> 5\%$	$> 5\%$			$> 5\%$	$> 5\%$	$> 5\%$	10^{-2}	$> 5\%$	$> 5\%$
	Exp	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$			$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$
	Avg	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$			$> 5\%$	$> 5\%$
	PDR	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	10^{-2}	$> 5\%$	10^{-3}		
COMET	**EIV**			10^{-3}	$> 5\%$	$> 5\%$	10^{-11}	$> 5\%$	$> 5\%$	10^{-4}	10^{-11}
	BCK	$> 5\%$	$> 5\%$			$> 5\%$	10^{-11}	$> 5\%$	$> 5\%$	$> 5\%$	10^{-11}
	Exp	$> 5\%$	$> 5\%$	10^{-2}	$> 5\%$			$> 5\%$	$> 5\%$	10^{-3}	10^{-10}
	Avg	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	10^{-11}			10^{-2}	10^{-11}
	PDR	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$	$> 5\%$		

Whitley statistical test, applied on the sets of $\hat{E}(I_{\epsilon^+}(R, A))$ values computed for each method.

Table 1 and 2 represents the comparison of the different selection methods for the $\hat{E}(I_{\epsilon^+})$ with the two different types of uncertainty: on objective vectors, and on decision variables. To compare the sets of runs, we use the Mann-Whitley statistical test, as described in [14]. The columns give the adjusted P value of the corresponding pairwise test that accounts for multiple testings; it equals to the lowest significance level for which the null-hypothesis (the medians are drawn from the same distribution) would still be rejected (with a significance level of 5%). A value under 5% shows that the method in the corresponding row is significantly better than the method in the corresponding column.

In many cases, the results are not significant in the bi-objective case, since the different approaches are similar, i.e. they use the same ϵ-indicator-based fitness assignment. But some conclusions could be extracted from table 1:

– The exponential approximation approach *Exp*, give worst results in many cases, excepted for *KUR* and *COMET* instances.
– *BCK* and *EIV* obtain similar results, which shows the efficiency of *BCK* to approximate *EIV* fitness assignment method.
– Uncertainty on objective vectors: in many cases, ϵ-indicator-based approaches *Avg*, *BCK* and *EIV* perform significantly better than Hughes selection mechanism *PDR*, especially for *COMET* and *ZDT*1 instances.
– Uncertainty on decision variables: *Avg* results are significantly worst than *EIV*, *BCK* and *PDR*, in many cases (problems *DTLZ*2, *ZDT*6 and *KUR*).

In table 2, we represent the results obtained for experimentations realized on *DTLZ*2 test function, with different number of objectives. This table shows a superior performance of *EIV*, *BCK* and *Exp* fitness assignment methods when

Table 2. Evaluation with several number of objectives to optimize: *DTLZ*2 test function, using the Mann-Whitley statistical test: P value, with noise on objective vectors (Z) and on decision vectors (X). A cell 10^{-a} corresponds to a significance level in the interval $[10^{-a}, 10^{-(a-1)}]$.

		EIV		**BCK**		**Exp**		**Avg**		**PDR**	
		Z	X	Z	X	Z	X	Z	X	Z	X
n=5	**EIV**			> 5%	> 5%	10^{-8}	10^{-9}	10^{-2}	10^{-7}	10^{-11}	10^{-11}
	BCK	> 5%	> 5%			10^{-9}	10^{-9}	10^{-2}	10^{-7}	10^{-11}	10^{-11}
	Exp	> 5%	> 5%	> 5%	> 5%			> 5%	> 5%	10^{-11}	10^{-10}
	Avg	> 5%	> 5%	> 5%	> 5%	10^{-6}	10^{-2}			10^{-11}	10^{-11}
	PDR	> 5%	> 5%	> 5%	> 5%	> 5%	> 5%	> 5%	> 5%		
n=10	**EIV**			> 5%	> 5%	10^{-2}	10^{-7}	10^{-2}	10^{-7}	10^{-11}	10^{-11}
	BCK	> 5%	> 5%			> 5%	10^{-8}	10^{-2}	10^{-8}	10^{-11}	10^{-11}
	Exp	> 5%	> 5%	> 5%	> 5%			> 5%	> 5%	10^{-11}	10^{-11}
	Avg	> 5%	> 5%	> 5%	> 5%	> 5%	> 5%			10^{-11}	10^{-11}
	PDR	> 5%	> 5%	> 5%	> 5%	> 5%	> 5%	> 5%	> 5%		

the number of objective functions is growing. Exp is dominated by EIV and/or BCK in several cases, especially for the 5 objectives instance.

5 Discussion

In this paper, we propose a method for handling uncertainty in indicator-based evolutionary algorithm. Our approach tries to make no assumption about distribution, bounds and general tendency of the uncertainty. We propose the algorithm EIV, which computes the exact expected value of ϵ-indicator. In order to apply this algorithm to environmental selection in EAs, we propose several algorithms which approximate the results obtained by EIV, which select the best possible solutions during environmental selection, according to the ϵ-indicator performance metric. We have made several experimentations. First, we consider the goal to minimize the loss in quality during environmental selection: BCK give a good approximation of EIV selection, which is more time consuming. Then, we made some experimentations on classical tests functions. Our proposed method give interesting results with an increasing number of objective functions. This are preliminary results. More experimentations are needed to evaluate the different approaches on different problems and uncertainties. The first experimentation done for environmental selection process, with different levels of uncertainties, number of objective functions and sample size, show that the quality of the selected individuals, according to the $I_{\epsilon+}$ indicator, is improved by the use of EIV or BCK fitness assignment method. We can expect that entire runs results will show the same tendency, but further experimentations has to be done. We also need to define new comparison test, which involve a set of expected objective vectors values, without knowledge about the true objective vector.

References

1. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Birmingham, UK (2004) 832–842
2. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Transactions on evolutionary computation **9** (2005) 303–317
3. Arnold, D.V.: A comparison of evolution strategies with other direct search methods in the presence of noise. Computational Optimization and Applications **24** (2003) 135–159
4. Horn, J., Nafpliotis, N.: Multiobjective optimization using the niched pareto genetic algorithm. Technical report, University of Illinois, Urbana-Champaign, Urbana, Illinois, USA (1993)
5. Hughes, E.: Evolutionary multi-objective ranking with uncertainty and noise. In: EMO'01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, London, UK, Springer-Verlag (2001) 329–343
6. Teich, J.: Pareto-front exploration with uncertain objectives. In: Conference on Evolutionary Multi-Criterion Optimization (EMO'01). Volume 1993 of Lecture Notes in Computer Science (LNCS). (2001) 314–328

7. Babbar, M., Lakshmikantha, A., Goldberg, D.E.: A modified NSGA-II to solve noisy multiobjective problems. In et al., E.C., ed.: Genetic and Evolutionary Computation Conference (GECCO'2003), late breaking papers. Volume 2723 of Lecture Notes in Computer Science., Chicago, Illinois, USA, Springer (2003) 21–27
8. Deb, K., Gupta, H.: Searching for robust pareto-optimal solutions in multi-objective optimization. In Coello Coello, C.A., Aguirre, A.H., Zitzler, E., eds.: Conference on Evolutionary Multi-Criterion Optimization (EMO'05). Volume 3410 of Lecture Notes in Computer Science (LNCS)., Guanajuato, Mexico, Springer-Verlag (2005) 150–164
9. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation **7** (2003) 117–132
10. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8** (2000) 173–195
11. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. Technical report, TIK Report Nr. 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich (2001)
12. Kursawe, F.: A variant of evolution strategies for vector optimization. In Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature, Springer (1991) 193–197
13. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. In Abraham, A., et al., eds.: Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications. Springer (2004) To appear.
14. Knowles, J.D., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastive multiobjective optimizers. Technical Report TIK-Report No. 214, Computer Engineering and Networks Laboratory, ETH Zurich (2005)

Fluctuating Crosstalk as a Source of Deterministic Noise and Its Effects on GA Scalability

Kumara Sastry[1], Paul Winward[1], David E. Goldberg[1], and Cláudio Lima[2]

[1] Illinois Genetic Algorithms Laboratory,
Department of General Engineering,
University of Illinois at Urbana-Champaign
[2] DEEI-FCT,
University of Algarve
{ksastry, winward, deg}@uiuc.edu, clima@ualg.pt

Abstract. This paper explores how fluctuating crosstalk in a deterministic fitness function introduces noise into genetic algorithms. We model fluctuating crosstalk or nonlinear interactions among building blocks via higher-order Walsh coefficients. The fluctuating crosstalk behaves like exogenous noise and can be handled by increasing the population size and run duration. This behavior holds until the strength of the crosstalk far exceeds the underlying fitness variance by a certain factor empirically observed. Our results also have implications for the relative performance of building-block-wise mutation over crossover.

1 Introduction

Recently, Sastry and Goldberg [1] presented an unbiased comparison between the computational costs associated with crossover in a selectorecombinative genetic algorithm (GA) to that of mutation. In that study, the mutation algorithm exploits its linkage knowledge to greedily change one building block (BB) at a time. In deterministic problems, the mutation algorithm outperforms the GA. However, in the presence of constant exogenous Gaussian noise, the situation flips as the selectorecombinative GA comes out on top.

One might wonder how these operators would fare in the presence of crosstalk, or what is sometimes referred to as epistasis. Goldberg [2] conjectures that one type of crosstalk, fluctuating crosstalk, can induce similar effects to explicit Gaussian noise, although still deterministic. He explains how a GA can converge to the global optimum if the same approaches are applied as if external noise was present: supply a larger population of individuals and allow more time for population convergence. Furthermore, the needed population and convergence time follow facetwise models derived for fitness functions with additive Gaussian noise when the crosstalk signal falls below a certain critical point. The purpose of this paper is to construct a bounding test function that demonstrates this effect when the fluctuation noise is varied from the nonexistent to the very high, and understand this in the light of recent theoretical decomposition of problem difficulty as pointed out elsewhere [2].

This paper is organized as follows. We first present a brief background to crosstalk in the context of a three-way decomposition to problem difficulty. In section 3, we describe how to represent fluctuating crosstalk through Walsh transformations and explain how many deterministic functions possess some form of fluctuating crosstalk. The next section compares our results to known models of population size, convergence time, and function evaluations. A threshold is observed for when fluctuating crosstalk deviates from these models based on the expected population size when crosstalk acts as a signal rather than noise. The paper concludes with a connection of our results to mutation and crossover performance and briefly addresses future work.

2 Crosstalk: A Facet of Problem Difficulty

Epistasis in GA literature refers to the nonlinear interactions among genes or sets of genes, and is widely known to be a contributing factor of GA-hardness. Historically, there have been three primary approaches to tackling epistasis [3]. The first approach relies on a priori measurements of problem difficulty which can either be direct or indirect measurements of epistasis. Davidor [4] first suggested that the amount of epistasis determines problem hardness for a GA. However, Naudts [5] observes that it is the distribution and structure of the epistasis that contributes to problem difficulty, and not only the amount of it. He shows these a priori measures are sensitive to non-linear scalings and require further theoretical development for robust and accurate measures. He introduces the concept of a site-wise optimization measure but careful probing reveals that it fails to correctly identify difficulty in several cases. A related area of these measurements is epistasis approximation through random sampling [4] but Naudts [5] finds random sampling a poor solution for approximating values of such measures. Heckendorn [6] derives local bitwise epistasis measures that scale well to overall difficulty for certain polynomial functions. He proposes techniques of argument centering and parity truncation to reduce epistasis for this class of functions. The second common approach is to use variants of the Factorized Distribution Algorithm (FDA) [7] — a probabilistic model building GA (PMBGA) [8] with multivariate interactions. Although such techniques directly confront epistasis, the FDA only guarantees optimal solutions for particular gene interactions [9] and requires prior information generally not available. The third approach calls for a change in problem representation but direct approaches [10] on the problem require a hefty amount of ingenuity and manipulation. Indirect representational approaches like the popular Gray Code "shifting" technique [11] show practical promise but the clarifying of the relationship between the need for shifting and epistasis remains to be done.

In this paper, we validate a fourth approach first proposed by Goldberg [2]. In discussing problem difficulty, Goldberg identified the three forms of crosstalk as a mapping to his three identified primary sources of problem difficulty: deception, scaling, and exogenous noise. Hence, a simple but effective way to optimize deterministic functions in the presence of crosstalk is to apply known techniques of handling deception, scaling, and exogenous noise. It becomes readily apparent of the mapping to deception and scaling, and these cases are well addressed in the literature. We explain the mapping between fluctuating crosstalk and exogenous noise with an example.

Consider the objective fitness function $f(x) = f_1(x_1x_2x_3x_4) + f_2(x_5x_6x_7) + f_3(x_8x_9)+f_4(x_5x_6x_7x_8x_9)$. This function has three BBs defined by bit ranges: x_1 to x_4, x_5 to x_7, and x_8 to x_9. Each bit range corresponds to a set of decision variables that are related to one another. We assume that the GA is aware of these substructures and will seek to find the values over these positions that maximize fitness. In a problem without epistasis (f_4), each BB is independent of each other and the global solution can be found by optimizing the sub-functions f_1, f_2, and f_3. Finding the correct BBs means finding the bit values that optimize such sub-functions. The concatenation of these BBs gives an optimal solution for f. In general, such substructures are not known ahead of time but in many cases are believed to exist and can be found in parallel through an array of GAs [2].

From this equation we can see how crosstalk or epistasis defined by f_4 refers to the nonlinear interaction among BBs. f_4 may be positive (reinforcing crosstalk), negative (punishing crosstalk), or a mix of positive and negative values. Goldberg [2] has shown that reinforcing crosstalk maps to scaling, and punishing crosstalk maps to scaling or deception, and that a competent GA that can handle scaling and deception can handle reinforcing and punishing crosstalk.

Now, suppose that instead of always punishing or rewarding the individual once the correct target bits are found, we add or subtract some positive weight w to the fitness function based on the parity over the target bits; $+w$ for even parity and $-w$ for odd parity. A natural question to consider is how drastic shifts in fitness because of a single bit change can be overcome during the course of BB decision making. Fortunately, two things act in our favor. First, the initial populations are assumed sufficiently randomized such that the target bits over BBs participating in the crosstalk are random. Hence, the net effect on fitness due to fluctuating crosstalk is zero since there are equal numbers of even-parity individuals as there are odd-parity individuals over those target bits. The second factor in our favor is the fact that ultimately, the GA will converge even with a possible selection stall. Thus, towards the end of the run fluctuation crosstalk behaves as either reinforcing crosstalk or punishing crosstalk and a GA that can handle those can effectively handle fluctuating crosstalk.

3 Principled Modeling of Fluctuating Crosstalk Through Walsh Coefficients

Having explained fluctuating crosstalk in the context of problem difficulty, we must now address how we model it in a principled way to account for any conceivable fluctuating crosstalk subfunction. This will be accomplished by representing the fitness function in another basis, the *partial-parity* or *Walsh basis*.

The Walsh basis is significant to GAs because it allows for bounding measures of deceptiveness and for faster schema-average fitness calculation. And in our case, it allows us to represent our fluctuating crosstalk example f_4 as a partial, signed sum of the Walsh coefficients. We begin with some notation and definitions.

For our discussion, we take the intuitive approach introduced by Goldberg [12]. Let $\mathbf{x} = x_l x_{l-1} \ldots x_2 x_1$ be the l-bit string representing the coding of the decision variables for an arbitrary individual. We now introduce the auxiliary string positions y_i that are mapped to from the bitwise string positions x_i for $i = 1, \ldots, l$ by:

$$y_i = \begin{cases} 1, & \text{if } x_i = 0 \\ -1, & \text{if } x_i = 1 \end{cases} .$$

This definition allows the following multiplication to act as an exclusive or operator (XOR). The j^{th} Walsh function $\psi_j(y)$ where $0 \leq j \leq 2^l - 1$ is calculated as:

$$\psi_j(y) = \prod_{i=1}^{l} y_i^{j_i}, \; y_i \in \{-1, 1\}$$

where j_i is the i^{th} bit of the binary representation of j and y is understood to be the transformed x using the auxiliary mapping.

In the canonical basis, fitness values are obtained by referencing a table of bitstrings and their objective fitness values. Bethke [13] showed that any fitness function $f(x)$ over a finite domain can be rewritten as a partial signed sum of the Walsh coefficients, given by

$$f(x) = \sum_{j=0}^{2^l-1} w_j \psi_j(x) \; .$$

For an arbitrary fitness function, we can imagine that some portion of the partial signed sum of the Walsh coefficients has direction - meaning that this smaller sum is acyclic in the limit and represents what the GA seeks to solve. The remaining portion fluctuates with possibly irregular periods and shapes - but not contributing to the overall direction of the function. This latter portion will likely involve higher-order Walsh coefficients since a higher order indicates that more bits interact with another (since more bits of the index are set). Consider the inclusion of the Walsh coefficient w_{2^l-1} for example. This means taking the parity of the entire string. It provides no direction but merely acts as a source of deterministic noise. Other coefficients could be included as part of this fluctuating crosstalk but for this paper we assume only a full parity.

4 Effect of Crosstalk on GA Scalability

Exogenous noise is a problem for GAs because it interferes with the decision making process as a GA tries to identify the dividing lines between building blocks and infer the superiority of one partition member over another. In this section, we compare exogenous noise effects to those wrought by deterministic noise and empirically validate model adherence. We present facetwise models of population, convergence time, and function evaluations when exogenous noise is present and examine at what point fluctuating crosstalk diverges from these models. However, we first need to introduce the test problem that was common to the three experiments.

4.1 Test Problem

To test the effects of deterministic noise on population size, convergence time, and function evaluations, our test problem consists of an on-average deceptive, concatenated trap function. The use of a trap function assumes a knowledge of linkage information but this well suits our situation since we want to focus on the effects of fluctuating crosstalk and assume that other factors are favorably constant. Such a function allows for efficient exchanging of BBs since each BB is independent of each other. The concatenated trap also allows for controlled testing of problem difficulty since both the order of the trap and number of traps play a role in determining problem difficulty. Each BB consists of a 5-bit trap function with a max fitness of 1 at 11111 and other fitness values linearly interpolated from $1-d$ to 0 as the number of 1's range from 0 to 4 where d is called the *signal*.

The Walsh coefficient w_{2^l-1} is then added to or subtracted from this temporary fitness based on the full parity of the string to obtain the final fitness. Modeling epistasis in this manner translates to the highest bitwise interaction possible where every bit interacts with every other bit. Note that the inclusion of an even number of BBs (m, ranging from 4 to 50) preserved optimal solutions under full parity although individual BBs were penalized. The GA utilized binary tournament selection ($s = 2$) and no mutation. We also chose uniform-BB crossover [1] to avoid disrupting discovered BBs. Our purpose in using this is to show that even in the best case when perfect crossover is used, deterministic noise exists and its effects of deterministic noise can be modeled by known models dealing with exogenous noise.

For each of the following plots, data was obtained by performing 30 *bisection runs* of 50 independent GA trials. In a single bisection run, the population size is adjusted after each set of 50 trials until the minimum population size is obtained that yields an average of $m-1$ correctly discovered BBs. The interested reader may refer to [14] for population size adjusting details. We then report the average of the averages over all bisection runs.

4.2 Population Size

One simple but effective way to overcome the negative effects of exogenous noise is to supply a larger population. Increasing the population size sufficiently ensures that in the initial generations the target BBs are present and discovered by the averaging out of the noise. It also assists in decision making over time so the

GA might reliably choose the best partition member. Practical population sizing bounds on selectorecombinative GAs using generation-wise modeling were given by Goldberg, Deb, and Clark [15]. Those early bounds provided a guarantee of good solution quality in a selectorecombinative GA with a sufficiently large population and have shown to well approximate solution quality in the more recent Bayesian Optimization Algorithm (BOA) [16]. They were known to be somewhat conservative for the typical selectorecombinative GA, however, and tighter bounds provided by Harik, Cantú-Paz, and Goldberg [17] are derived from the gambler's ruin problem which considers the accumulated correctness of deciding between partition members. This model, which also accounts for exogenous noise, is given by:

$$n = -\frac{\sqrt{\pi}}{2d} 2^k \log(\alpha) \sqrt{\sigma_f^2 + \sigma_N^2} \tag{1}$$

where d is the *signal* or difference in fitness between the best and second best individuals, k is the BB size, α is the error tolerance, σ_f^2 is the fitness function variance, and σ_N^2 is the noise variance.

Without noise, this reduces to

$$n_0 = -\frac{\sqrt{\pi}}{2d} 2^k \log(\alpha) \sigma_f \ . \tag{2}$$

Dividing 1 by 2 and letting $\sigma_N^2 = w_{2^l-1}^2$ and $\sigma_f^2 = m\sigma_{BB}^2$, where σ_{BB}^2 is the 5-bit trap variance, reveals an important population size ratio:

$$n_r = \frac{n}{n_0} = \sqrt{1 + \frac{w_{2^l-1}^2}{m\sigma_{BB}^2}} \ . \tag{3}$$

The above model was specifically derived with exogenous noise in mind but what effect on required population size does fluctuating crosstalk have? When the crosstalk signal is low, we would expect the GA to first solve BBs for the fitness function, since solving these will yield higher marginal contributions than those BBs of the crosstalk. The crosstalk merely interferes with the decision making, acting as a source of deterministic noise. As the crosstalk signal increases, the problem shifts to a parity-dominated one whereby individuals with even parity are first discovered and then varied to find the secondary benefits of solving the trap function. In this sense the directional function is actually perturbing the crosstalk. Once on the parity-dominated side, the required population will level off since the population is large enough for the GA to find the optimal or near-optimal solution and supplying a larger crosstalk signal only creates an effective reduction in the marginal value of solving BBs from the new perturbation source.

Figure 1 illustrates these principles clearly. Before the parity-dominated point but for a given crosstalk signal, required population size grows as problem size m grows. More building blocks induce higher collateral noise since more BBs incur larger numbers of schemata to consider, and the resulting changes to a single BB due to crossover are obfuscated by the simultaneous variations among

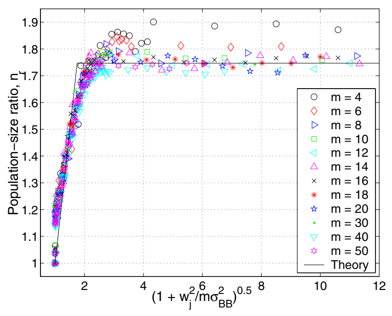


Fig. 1. Population size requirements for optimal convergence when fluctuating crosstalk is present ($w_j = w_{2^l-1}$) and problem size m varies from 4 to 50 BBs. Initially, the effects of the crosstalk follow the model (rising bolded line) of exogenous noise closely but then level off when the problem becomes parity-dominated. The flat bolded line is the average required population for all problem sizes tested.

other crossover-induced BB changes. What is not as intuitive though is why the population size tends to level off where it does.

This can be explained by what we term the *parity-induced filtering effect*. We begin by considering the behavior of the GA as it nears the situation described by the middle of the curve. Here, the partial parity function plays a major role in BB processing just as the previous deterministic function still does. For full l-bit parity and early on in the GA processing, half of the population will lose its market share since half will be penalized for its odd number of ones. For the other half, the GA is processing the deterministic function. Hence, it requires roughly twice the population needed when no noise is present ($n_r \approx 2$). This effect is compounded by the subsequent generation of individuals. Of course, this only applies for the initial generations where schema diversity is high. It should also be remembered that the directional function still plays a nontrivial role at this critical point and we would not expect a true doubling of the population.

If we assume an upperbound of a doubling in required population size, as empirically observed, we can estimate the critical point for the corresponding crosstalk signal:

$$n_r = 2 = \sqrt{1 + \frac{w_{2^l-1}^2}{m\sigma_{BB}^2}} \tag{4}$$

and hence

$$w_{2^l-1}^2 = 3m\sigma_{BB}^2 \ . \tag{5}$$

Now, recall equation 3:

$$n_r = \frac{n}{n_0} = \sqrt{1 + \frac{w_{2^l-1}^2}{m\sigma_{BB}^2}} \ .$$

This represents the ratio of the required population size for some fitness function with noise to the required population size for the same function without noise. Note that d has entirely dropped out of the equation. From Fig. 2, we see that d actually has a small influence on the population enlargement factor due to noise and recognize that our model does not capture this small difference.

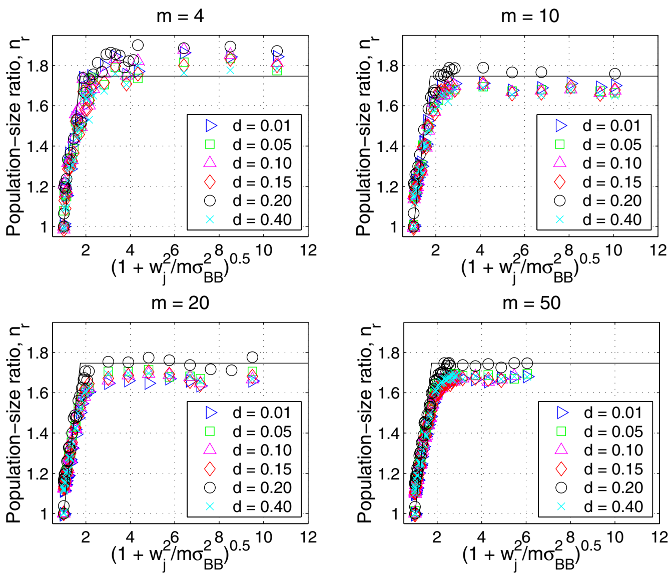


Fig. 2. Population size requirements for optimal convergence when fluctuating crosstalk is present $(w_j = w_{2^l-1})$

4.3 Convergence Time

Our scalability analysis would not be complete without considering the effects of deterministic noise on convergence time. With a little reasoning we see that the filtering effect elongates convergence time similarly to growing the required population size. This may be thought of as a bump in selection pressure since with s-tournament selection the GA now needs $2s$ individuals to have the same *quality of choices* in selection. Individuals of the wrong parity are immediately discarded, and can only be chosen if all s individuals are placed into the selection pool.

Various convergence time models have been employed over the years. We forgo the development of convergence time models here but the interested reader may refer to [14] and [2] for such cronologies. For our discussion, we note that convergence time models based on quantitative genetics [18, 19] are proving especially useful [20, 21, 22, 23, 24, 25, 26, 2]. A facetwise model using selection pressure in the presence of exogenous noise that we employ here is that of Miller and Goldberg (1995, 1996):

$$t_c = \frac{\pi\sqrt{l}}{2I}\sqrt{1+\frac{\sigma_N^2}{\sigma_f^2}} \tag{6}$$

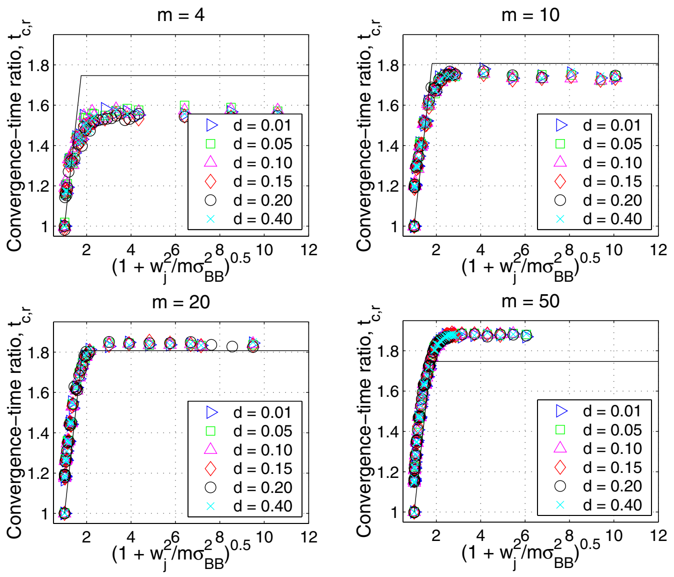


Fig. 3. Convergence time requirements for optimal convergence when fluctuating crosstalk is present ($w_j = w_{2^l-1}$)

where I is the *selection intensity* [18, 24, 27]. As we saw with population size, the ratio of the convergence time under noise to that without noise does not depend on the signal. We start by considering the convergence time needed when noise is absent:

$$t_{c0} = \frac{\pi\sqrt{l}}{2I} \quad . \tag{7}$$

By casting equation 6 in terms of t_{c0}, and then dividing both sides by t_{c0}, we obtain the convergence time ratio:

$$t_{c,r} = \frac{t_c}{t_{c0}} = \sqrt{1 + \frac{w_{2^l-1}^2}{m\sigma_{BB}^2}} \quad . \tag{8}$$

Note again that the convergence time doesn't depend on the trap signal d. We observe from Fig. 3 that the predicted plots follow the model well for varying signals and building blocks. In comparing the predicted plots, it is the sloping solid line that represents the prediction. For $m = 4$, we see that convergence time follows precisely what is predicted but reaches the critical threshold earlier than the average of the time-convergence runs. This can be attributed to the fact that for small m, fewer BBs means fewer simultaneous variations among other crossover-induced BB changes and is hence more sensitive to larger Walsh coefficients. Sensitivity reduces for larger m, and we see that the critical threshold levels out just under a $t_{c,r}$ of 2.

4.4 Function Evaluations

The number of function evaluations needed to reliably find the optimal solution is a product of convergence time and population size. This measure of time is the true test of performance and based on our results of population size and convergence time, we expect results to follow the model initially until the critical point given by equation 5. Afterwards, the parity-dominated side behaves as explained previously. Figure 4 confirms this but also gives way to some immediate conclusions.

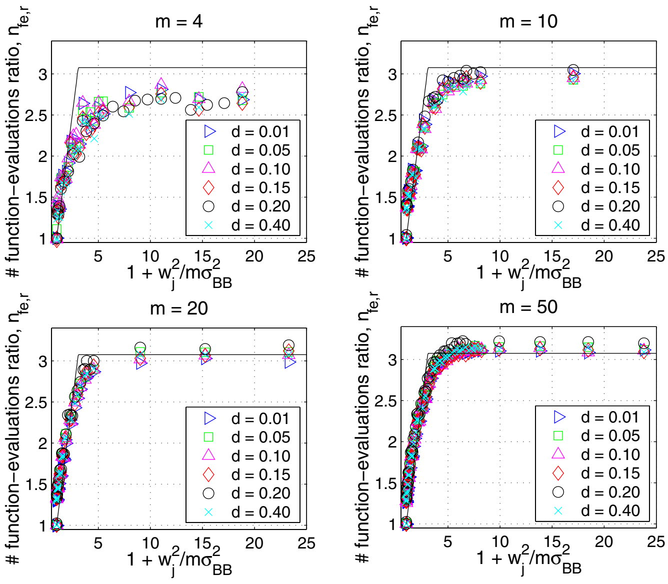


Fig. 4. Functional evaluation requirements for optimal convergence when fluctuating crosstalk is present ($w_j = w_{2^l-1}$)

5 Future Work

Future work includes modeling fluctuating crosstalk using smaller order Walsh coefficients and determining a more precise transition point between crosstalk-as-noise and crosstalk-as-signal. Another useful direction is to consider various forms of epistasis such as if only a portion of the entire chromosome is used to determine the parity of the individual. Such epistasis may be uniformly distributed as parity bits within each BB, confined to entire BBs, or be a mixture of both. Of course, bits may also be involved in multiple parity evaluations. We seek to consider these effects on GA scalability and under what conditions fluctuating crosstalk behaves like exogenous noise.

6 Summary and Conclusions

We have illustrated the introduction of noise in a deterministic fitness function via fluctuating crosstalk. We modeled fluctuating crosstalk with higher-order Walsh coefficients and showed that fluctuating crosstalk behaves like additive exogenous noise until the crosstalk variance far exceeds the underlying fitness variance by a certain threshold we empirically observe. While the crosstalk behaves similarly to external noise, its effects can be handled in a similar manner by increasing the population size and run duration.

Returning to where we started, the question that motivated this study was a prior performance comparison [1] between mutation and crossover with and without exogenous noise. We have shown that fluctuating crosstalk acts as a source of deterministic noise and affects GA performance similarly to exogenous noise. For a precise discussion of when mutation is preferred over the selectorecombinative or vice versa, the reader should consult the aforementioned work.

References

1. Sastry, K., Goldberg, D.E.: Let's get Ready to Rumble: Crossover Versus Mutation Head to Head. Proceedings of the 2004 Genetic and Evolutionary Computation Conference **2** (2004) 126–137 Also IlliGAL Report No. 2004005.
2. Goldberg, D.E.: Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Acadamic Publishers, Boston, MA (2002)
3. Kumar, V.: Tackling Epistasis: A Survey of Measures and Techniques. Assignment from an advanced GEC course taught at UIUC by D. E. Goldberg (2002)
4. Davidor, Y.: Epistasis Variance: A Viewpoint on GA-hardness. foga91 (1991) 23–35
5. Naudts, B., Kallel, L.: Some Facts about so-called GA-hardness Measures. Tech. Rep. No. 379, Ecole Polytechnique, CMAP, France (1998)
6. Heckendorn, R.B., Whitley, D.: Predicting Epistasis from Mathematical Models. Evolutionary Computation **7**(1) (1999) 69–101
7. Mühlenbein, H., Mahnig, T., Rodriguez, A.O.: Schemata, Distributions and Graphical Models in Evolutionary Optimization. Journal of Heuristics **5** (1999) 215–247
8. Pelikan, M., Goldberg, D.E., Lobo, F.G.: A Survey of Optimization by Building and Using Probabilistic Models. Comput. Optim. Appl. **21**(1) (2002) 5–20
9. Lauritzen, S.L.: Graphical Models. Oxford University Press (1998)
10. Beasley, D., Bull, D.R., Martin, R.R.: Reducing Epistasis in Combinatorial Problems by Expansive Coding. In: ICGA. (1993) 400–407
11. Barbulescu, L., Watson, J.P., Whitley, L.D.: Dynamic Representations and Escaping Local Optima: Improving Genetic Algorithms and Local Search. In: AAAI/IAAI. (2000) 879–884
12. Goldberg, D.E.: Genetic Algorithms and Walsh Functions: Part I, a Gentle Introduction. Complex Systems **3**(2) (1989) 129–152 (Also TCGA Report 88006).
13. Bethke, A.D.: Genetic Algorithms as Function Optimizers. PhD thesis, The University of Michigan (1981)
14. Sastry, K.: Evaluation-Relaxation Schemes for Genetic and Evolutionary Algorithms. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL (2001) (Also IlliGAL Report No. 2002004).

15. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic Algorithms, Noise, and the Sizing of Populations. Complex Systems **6** (1992) 333–362 (Also IlliGAL Report No. 91010).
16. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: Linkage Learning, Estimation Distribution, and Bayesian Networks. Evolutionary Computation **8**(3) (2000) 314–341 (Also IlliGAL Report No. 98013).
17. Harik, G., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. Evolutionary Computation **7**(3) (1999) 231–253 (Also IlliGAL Report No. 96004).
18. Bulmer, M.G.: The Mathematical Theory of Quantitative Genetics. Oxford University Press, Oxford (1985)
19. Falconer, D.S.: Introduction to Quantitative Genetics. Third edn. John Wiley and Sons, New York, NY, USA; London, UK; Sydney, Australia (1989)
20. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm: I. Continous Parameter Optimization. Evolutionary Computation **1**(1) (1993) 25–49
21. Mühlenbein, H., Schlierkamp-Voosen, D.: The Science of Breeding and its Application to the Breeder Genetic Algorithm (BGA). Evolutionary Computation **1**(4) (1994) 335–360
22. Thierens, D., Goldberg, D.E.: Convergence Models of Genetic Algorithm Selection Schemes. Parallel Problem Solving from Nature **3** (1994) 116–121
23. Thierens, D., Goldberg, D.E.: Elitist Recombination: An Integrated Selection Recombination GA. Proceedings of the First IEEE Conference on Evolutionary Computation (1994) 508–512
24. Bäck, T.: Generalized Convergence Models for Tournament—and (μ, λ)—Selection. Proceedings of the Sixth International Conference on Genetic Algorithms (1995) 2–8
25. Miller, B.L., Goldberg, D.E.: Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise. Evolutionary Computation **4**(2) (1996) 113–131 (Also IlliGAL Report No. 95009).
26. Voigt, H.M., Mühlenbein, H., Schlierkamp-Voosen, D.: The Response to Selection Equation for Skew Fitness Distributions. Proceedings of the International Conference on Evolutionary Computation (1996) 820–825
27. Blickle, T., Thiele, L.: A Mathematical Analysis of Tournament Selection. Proceedings of the Sixth International Conference on Genetic Algorithms (1995) 9–16

Integrating Techniques from Statistical Ranking into Evolutionary Algorithms

Christian Schmidt[1], Jürgen Branke[1], and Stephen E. Chick[2]

[1] Institute AIFB, University of Karlsruhe, Germany
csc@aifb.uni-karlsruhe.de, branke@aifb.uni-karlsruhe.de
[2] INSEAD, Technology and Operations Management Area,
Fontainebleu, France
stephen.chick@insead.edu

Abstract. Many practical optimization problems are subject to uncertain fitness evaluations. One way to reduce the noise is to average over multiple samples of the fitness function in order to evaluate a single individual. This paper proposes a general way to integrate statistical ranking and selection procedures into evolutionary algorithms. The proposed procedure focuses sampling on those individuals that are crucial for the evolutionary algorithm, and distributes samples in a way that efficiently reduces uncertainty. The goal is to drastically reduce the number of evaluations required for a proper operation of the evolutionary algorithm in noisy environments.

Keywords: Evolutionary algorithm, noise, ranking, selection.

1 Introduction

In many practical optimization problems, a solution's fitness is noisy, i.e. it can not be determined accurately and has to be considered a random variable. The sources for noise can be manifold, including optimization based on randomized simulations, measurement errors, stochastic sampling, and interaction with users.

Generally, noise is considered a major challenge for optimization, as it makes it difficult to decide which of two solutions is better, and thus to drive the search reliably towards the more promising areas of the search space. The effect of noise on the performance of evolutionary algorithms (EAs) has been investigated in several papers, and EAs are generally considered to be quite robust with respect to noise, see e.g. [1, 3, 14]. A recent survey on this topic is [16].

For most noisy optimization problems, the uncertainty in fitness evaluation can be reduced by sampling a solution's fitness several times and using the average as estimate for the true mean fitness. Sampling n times scales the standard deviation of the estimator of the mean by a factor of $1/\sqrt{n}$, but increases the computational time by a factor of n. This is a critical tradeoff trade-off: either one can use relatively exact estimates but only run the algorithm for a small number of iterations (because a single fitness estimate requires many evaluations), or one can let the algorithm work with relatively crude fitness estimates, but allow for more iterations (as each estimate requires less effort).

This paper presents a new way to integrate statistical ranking and selection techniques into EAs in order to improve their performance in noisy environments. Ranking and selection addresses the question of identifying the individual with the true best mean out of a given (finite) set of individuals by sampling the individuals' fitnesses. Thereby, it is attempted to achieve a desired selection quality (e.g. probability of correct selection) with a minimal number of samples. We propose a framework that tightly integrates an efficient statistical selection procedure with an EA, allowing it to focus on those pairwise comparisons that are crucial for the EA's operation.

Section 2 briefly surveys related work. Section 3 introduces a statistical selection technique, $\mathcal{OCBA}$, that is used in the EA. Section 4 examines the information requirement of the EA. Section 5 explains how $\mathcal{OCBA}$ is adapted to generate that information efficiently. Section 6 gives preliminary empirical results.

2 Related Work

There have been some earlier attempts to integrate ranking and selection techniques into EAs. Stagge [17] considered a (μ, λ) or $(\mu+\lambda)$ evolution strategy and suggested that the sample size should be based on an individual's probability to be among the μ best ones that will be selected. Hedlund and Mollaghasemi [15] use an indifference-zone selection procedure to select the best m out of k individuals within a genetic algorithm.

For tournament selection, Branke et al. [8, 9] and Cantu-Paz [11] use sequential sampling techniques to reduce the number of samples to the minimum required to discriminate between individuals in a tournament. Adaptive sampling strategies have also been examined for situations where the noise strength varies over space [13]. Boesel [4] argues that for linear ranking selection, it is sufficient to group individuals of similar quality into one rank, and a corresponding mechanism is proposed.

To "clean up" after optimization (to identify the best, with high probability, of all visited solutions), Boesel et al. [5] uses a ranking and selection procedure after the EA has finished. Recently, Buchholz and Thümmler [10] used a statistical selection technique for selection in a $\mu + \lambda$ strategy as well as to maintain a pool of promising candidates throughout the run from which at the end the final solution is selected.

None of the above works provides the general framework and tight integration of selection algorithms and EAs that is suggested here.

A comprehensive overview and extensive comparison of different selection procedures can be found in [7], which concludes that $\mathcal{OCBA}$ together with an appropriate stopping rule is among the best performing statistical selection procedures. Our approach is based on $\mathcal{OCBA}$ which shall be described next.

3 Optimal Computing Budget Allocation

The Optimal Computing Budget Allocation ($\mathcal{OCBA}$) [12] is a sequential ranking and selection procedure that is based on Bayesian statistics. The $\mathcal{OCBA}$ was improved with flexible stopping rules in [7], and was shown to be one of the most efficient selection procedures. We first define the sampling assumptions that were used to derive the procedure, then describe the procedure itself.

Let X_{ij} be a random variable whose realization x_{ij} is the output of the j-th evaluation of individual i, for $i = 1, \ldots, k$ and $j = 1, 2, \ldots$. Let w_i and σ_i^2 be the unknown mean and variance of the evaluated individual i, and let $w_{[1]} \leq w_{[2]} \leq \ldots \leq w_{[k]}$ be the ordered means. In practice, the ordering $[\cdot]$ is unknown, and the best individual, individual $[k]$, is to be identified with fitness sampling. $\mathcal{OCBA}$ assumes that simulation output is independent and normally distributed, *conditional* on w_i and σ_i^2, for $i = 1, \ldots, k$. Although this normality assumption is not always valid, it is often possible to batch a number of evaluations so that normality is approximately satisfied.

Let n_i be the number of replications for individual i run so far. Let $\bar{x}_i = \sum_{j=1}^{n_i} x_{ij}/n_i$ be the sample mean and $\hat{\sigma}_i^2 = \sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2/(n_i - 1)$ be the sample variance. Let $\bar{x}_{(1)} \leq \bar{x}_{(2)} \leq \ldots \leq \bar{x}_{(k)}$ be the ordering of the sample means based on all replications seen so far. Equality occurs with probability 0 in contexts of interest here. The quantities n_i, $\bar{x}_i, \hat{\sigma}_i^2$ and (i) may change as more replications are observed.

The standard selection problem is to identify the best of the k individuals, where 'best' means the largest expected fitness. From a Bayesian perspective the means W_i are $\mathtt{St}\left(\bar{x}_i, n_i/\hat{\sigma}_i^2, n_i - 1\right)$ distributed (assuming a non-informative prior), where $\mathtt{St}\,(m, \kappa, \nu)$ is a Student-distribution with mean m, precision κ and ν degrees of freedom. Upper case is used for the random variable, and lower case is used for the (as yet unknown) realization. Given the data $\mathcal{E}$ seen so far, the probability $\mathrm{PCS}_{\mathrm{Bayes}}$ that the individual with the best observed mean, individual (k), is really the best individual, $[k]$, can be approximated using the Slepian inequality and Welch approximations:

$$\begin{aligned} \mathrm{PCS}_{\mathrm{Bayes}} &\overset{\mathrm{def}}{=} \Pr\left(W_{(k)} \geq \max_{i \neq (k)} W_i \,|\, \mathcal{E}\right) \\ &\geq \prod_{i \neq (k)} \Pr\left(W_{(k)} \geq W_i \,|\, \mathcal{E}\right) \\ &\approx \prod_{i \neq (k)} \Phi_{\nu_{(k)i}}(d_{(k)i}/s_{(k)i}) \overset{\mathrm{def}}{=} \mathrm{PCS}_{\mathrm{Slep}}, \end{aligned}$$

with $\Phi_\nu(\cdot)$ the cummulative distribution function of Student's distribution, ν_{ij} the degrees of freedom by Welch's approximation, $d_{ij} = \bar{x}_i - \bar{x}_j$ the observed difference and $s_{ij} = \sqrt{\hat{\sigma}_i^2/n_i + \hat{\sigma}_j^2/n_j}$ the variance of the difference of the estimated means.

For most practical problems, it is not so important to identify the really best solution. Instead, it is sufficient to find a solution close to the optimum. This can be reflected by introducing an "indifference zone" δ^*. Then, $\mathrm{PCS}_{\mathrm{Slep}}$ is replaced by the probability of *good* selection,

$$\mathrm{PGS}_{\mathrm{Slep},\delta^*} \overset{\mathrm{def}}{=} \prod_{i \neq (k)} \Phi_{\nu_{(k)i}}\left((d_{(k)i} + \delta^*)/s_{(k)i}\right)$$

The $\mathcal{OCBA}$ variant working with $\text{PGS}_{\text{Slep},\delta^*}$ as quality goal is denoted $\mathcal{OCBA}_{\delta^*}$.

$\mathcal{OCBA}_{\delta^*}$ attempts to optimize $\text{PGS}_{\text{Slep},\delta^*}$ by greedily and iteratively allocating additional evaluations to individuals where it promises the largest improvement, assuming that the means do not change and the standard error is scaled back appropriately. More specifically, in a first stage of sampling, $\mathcal{OCBA}_{\delta^*}$ evaluates the fitness function n_0 times per individual. In each subsequent sequential stage, τ additional evaluations are given to one individual, and none to the others. The ith individual is selected in a given stage if it maximizes $\text{PGS}_{\text{Slep},\delta^*}$ when the distribution for the ith unknown mean is changed from $\texttt{St}\left(\bar{x}_i, n_i/\hat{\sigma}_i^2, n_i - 1\right)$ to

$$\tilde{W}_i \sim \texttt{St}\left(\bar{x}_i, (n_i + \tau)/\hat{\sigma}_i^2, n_i - 1 + \tau\right)$$

The $\mathcal{OCBA}_{\delta^*}$ stops sampling when $\text{PGS}_{\text{Slep},\delta^*}$ exceeds a prespecified probability of good selection level, $\text{PGS}_{\text{Slep},\delta^*} \geq 1 - \alpha^*$. See [12] for early work and [7] for a full specification of $\mathcal{OCBA}_{\delta^*}$.

4 Order Information Required by EAs

Looking at an EA's typical iteration (Figure 1), there are two steps which are affected by noise: In the *selection* step, better fit individuals are assigned a higher probability to be selected as mating partner. In the *replacement* step[1], the set of new and old individuals is reduced to the usual population size by removing some individuals depending on age and fitness.

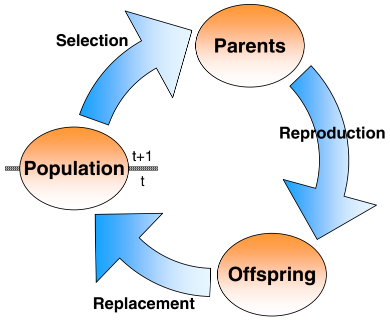


Fig. 1. Loop of an evolutionary algorithm

For deterministic fitness functions, the EA has access to the complete ordering of the combined set of individuals in the old population and the generated offspring population. That is, for any pair of individuals, it can determine with certainty which one is better. However, only parts of this information is actually

[1] This is often also called "environmental selection".

used within an EA generation. Therefore, in order to make the sampling more efficient, in this section we examine exactly what information is necessary for the operation of an EA. In Section 5, we then show how $\mathcal{OCBA}_{\delta^*}$ can be adapted to efficiently generate just this key information.

Different EA variants require different kinds of order information. We first describe different replacement and selection strategies. We then show how they can be combined into some typical algorithm variants.

We assume that the population size is μ, and λ offspring are generated in each iteration. We consider three orderings based on the observed means:

- $()_P$ denotes the ordering from worst to best of the individuals in the old population, i.e. $\bar{x}_{(1)_P} \leq \bar{x}_{(2)_P} \leq \ldots \leq \bar{x}_{(\mu)_P}$.
- $()_O$ denotes the ordering from worst to best of the individuals in the offspring population, i.e. $\bar{x}_{(1)_O} \leq \bar{x}_{(2)_O} \leq \ldots \leq \bar{x}_{(\lambda)_O}$.
- $()$ denotes the ordering from worst to best in the combined old and offspring populations, i.e. $\bar{x}_{(1)} \leq \bar{x}_{(2)} \leq \ldots \leq \bar{x}_{(\mu+\lambda)}$.

The order information required is specified as a set $\mathcal{C}$ containing pairs of individuals $\langle i, j \rangle$ that need to be compared.

If $\langle (i), (j) \rangle \in \mathcal{C}$ and $i > j$, it means that the EA will operate under the assumption that the individual on rank i is better than the individual on rank j, or $w_{(i)} > w_{(j)}$. Note that the pairs of individuals to be included in $\mathcal{C}$ may depend on the current ordering according to observed sample means, which may change during the sampling procedure.

4.1 Replacement Strategies

The replacement step determines which μ individuals, out of the $\mu+\lambda$ individuals from the old and offspring populations, survive to the next generation to form the new population.

In **generational replacement**, the old population is completely replaced by μ offspring, and no fitness information is needed. However, this is often combined with **elitism**, which means that only $\mu - 1$ offspring are generated, and the best individual from the old population is transfered to the new population. In this case, the algorithm needs to make sure that the (perceived) best individual in the old population is really better than all the others in the old population. In the notation introduced above, this means

$$\mathcal{C} \leftarrow \bigcup_{i=1}^{\mu-1} \{\langle (\mu)_P, (i)_P \rangle\}. \quad (1)$$

In a **steady state** EA, in each iteration a single offspring is generated and replaces the worst individual in the population. This only requires to identify the old population's worst individual, i.e.

$$\mathcal{C} \leftarrow \bigcup_{i=2}^{\mu} \{\langle (i)_P, (1)_P \rangle\} \quad (2)$$

In the (μ, λ) **replacement** typically used in evolution strategies, the best μ out of the λ children are selected as new population, i.e. the algorithm wants to ensure that individuals $(\lambda-\mu+1)_O, (\lambda-\mu+2)_O, \dots (\lambda)_O$ are really better than individuals $(1)_O, (2)_O, \dots (\lambda-\mu)_O$, or

$$\mathcal{C} \leftarrow \bigcup_{i=\lambda-\mu+1}^{\lambda} \{\langle (i)_O, (1)_O \rangle, \langle (i)_O, (2)_O \rangle, \dots, \langle (i)_O, (\lambda-\mu)_O \rangle\}\,. \tag{3}$$

For a $(\mu + \lambda)$ **replacement** where the new population is formed by the μ best of the $\mu + \lambda$ individuals from the old and offspring population, set

$$\mathcal{C} \leftarrow \bigcup_{i=\lambda+1}^{\lambda+\mu} \{\langle (i), (1) \rangle, \langle (i), (2) \rangle, \dots, \langle (i), (\lambda) \rangle\}\,. \tag{4}$$

4.2 Mating Selection

In the EA literature, different methods have been proposed to select, from the population, the parents that generate a child. Among the most popular are **linear ranking selection** and tournament selection, which generate the same expected probability for an individual to be selected. For standard **tournament selection**, two individuals i and j are chosen randomly from the population, and the better one is selected. Thus, for a single selection step, order information on only these two individuals is required:

$$\mathcal{C} \leftarrow \{\langle i, j \rangle\} \tag{5}$$

Evolution strategies usually use **random mating selection** which does not require any fitness information. Another popular selection strategy is **fitness proportional selection**. Since it is based on relative fitnesses rather than ranks, our method can not be applied here. However, fitness proportional selection is known to have some scaling and convergence issues, and generally rank-based selection methods are preferred anyway.

4.3 Combining Mating and Replacement

A particular EA uses a combination of selection and replacement strategies. These two steps are performed one after the other[2], and thus can be considered together with respect to the sampling procedure. This means that $\mathcal{C}$ contains the union of all pairs of individuals as required for the chosen selection and replacement strategies, with one peculiarity: Because selection is done after replacement, one has to make sure that the comparisons required within e.g. a tournament selection are also surviving the replacement step. This can be easily

[2] If the evolutionary cycle is broken up after reproduction, selection immediately follows replacement.

achieved by selecting ranks instead of individuals in the selection step. For example, for a tournament selection, instead of selecting the better of individuals i and j, one can select the better of individuals (p) and (q), where p and q are ranks in the surviving population.

For example, consider a simple steady-state EA with population size $\mu = 4$, and one offspring generated. The two individuals for mating are selected by two tournaments. Then, we need to insure that the worst individual is removed by replacement, i.e. $\mathcal{C} \leftarrow \{\langle (2)_P, (1)_P \rangle, \langle (3)_P, (1)_P \rangle, \langle (4)_P, (1)_P \rangle\}$. Furthermore, we need two random pairs of individuals for tournament, e.g. $\mathcal{C} \leftarrow \mathcal{C} \cup \{\langle (3), (2) \rangle, \{\langle (5), (3) \rangle\}$. Overall, this means only information about the relative order of $\mu + 1$ pairs of individuals needs to be correct, as opposed to the $\mu(\mu+1)/2 = 10$ pairs for a complete ordering. The savings increase with increasing population size. For e.g. $\mu = 20$, only 21 pairs of individuals need to be compared, as opposed to 210 for a complete ordering.

Additional savings in an EA setting stem from the fact that individuals that survived from the previous generation have already been re-evaluated. $\mathcal{OCBA}_{\delta^*}$ can make full use of this information. Furthermore, because it is likely that individuals surviving to the next iteration have been sampled more often than those that are killed off, there is an additional advantage of using $\mathcal{OCBA}_{\delta^*}$ compared to using a fixed sample size [2].

5 Using $\mathcal{OCBA}_{\delta^*}$ to Generate Order Information

One of $\mathcal{OCBA}_{\delta^*}$'s advantages is its flexibility. It can be easily adapted to not only select the best out of a given set of individuals, but for arbitrary quality measures. To integrate it into an EA, we want to make sure that the EA operates "correctly", meaning that the order relations required by the EA have been determined correctly with a sufficiently high probability. Section 4 explained how to determine the required set of comparisons $\mathcal{C}$. As a quality criterion, we define the probability of good generation ($\mathrm{PGG}_{\mathrm{Bayes}}$) as probability that all pairwise comparisons in $\mathcal{C}$ are correct. The following equation approximates the probability that for all pairs in $\mathcal{C}$, the individual with the higher observed rank also has a higher true mean value. It assumes rank $i >$ rank j for all $\langle i, j \rangle \in \mathcal{C}$. If rank $j >$ rank i, simply replace $\langle i, j \rangle$ by $\langle j, i \rangle$ in $\mathcal{C}$ before calculation.

$$\begin{aligned} \mathrm{PGG}_{\mathrm{Bayes}} &\overset{\mathrm{def}}{=} \Pr\left(\bigwedge_{\langle i,j \rangle \in \mathcal{C}} W_i + \delta^* > W_j \,|\, \mathcal{E} \right) \\ &\geq \prod_{\langle i,j \rangle \in \mathcal{C}} \Pr\left(W_i + \delta^* > W_j \,|\, \mathcal{E}\right) \\ &\approx \prod_{\langle i,j \rangle \in \mathcal{C}} \Phi_{\nu_{ij}}((\delta^* + d_{ij})/s_{ij}) = \mathrm{PGG}_{\mathrm{Slep},\delta^*} \end{aligned} \tag{6}$$

Here, the $\mathcal{OCBA}_{\delta^*}$ uses $\mathrm{PGG}_{\mathrm{Slep},\delta^*}$ as goal function just as it does with $\mathrm{PGS}_{\mathrm{Slep},\delta^*}$. It automatically and efficiently allocates samples to individuals until a desired goal $\mathrm{PGG}_{\mathrm{Slep},\delta^*} > 1 - \alpha^*$ is obtained.

The resulting EA using $\mathcal{OCBA}_{\delta^*}$ needs α^* as input:

Procedure $\mathcal{OCBA}_{\delta^*}^{EA}(\alpha^\star)$

1. Evaluate each new individual n_0 times (old individuals have already been sampled at least n_0 times). Estimate the ranks by ordering the individuals based on the observed mean values.
2. Determine $\mathcal{C}$: initialize $\mathcal{C} \leftarrow \emptyset$ and add comparisons from the operators as described in Sections 4.1–4.2.
3. WHILE the observed results are not sufficiently sure ($\mathrm{PGG}_{\mathrm{Slep},\delta^*} < 1 - \alpha^\star$) DO
   (a) Allocate new evaluations to the individuals according to the $\mathcal{OCBA}_{\delta^*}$-allocation rule.
   (b) If ranks have changed from the previous iteration of the ranking procedure, update $\mathcal{C}$: initialize $\mathcal{C} \leftarrow \emptyset$ and add comparisons from the operators as described in Sections 4.1–4.2.

$\mathcal{OCBA}_{\delta^*}^{\mathrm{EA}}$ is called in every iteration of the EA after the offspring has been generated and before replacement. Then, the EA proceeds simply using the ordering given by the sample means.

The number of samples taken by $\mathcal{OCBA}_{\delta^*}^{\mathrm{EA}}$ depends on the problem configuration and the settings of α^* and δ^*. It may be useful to vary α^* over time, as higher accuracy may be needed towards the early and late phases of the algorithm (cf. [6]).

6 Empirical Evaluation

We empirically compare different sampling mechanisms based on their performance on a single iteration. The proposed integration of ranking and selection with EAs needs a more exhaustive evaluation in future work.

We stochastically generated $k = 10$ individuals with means distributed according to the negative of an exponential distribution with mean 1 and variances distributed according to an inverted gamma distribution with $\alpha = 100$ and $\beta = 99$ (Figure 2). Such a distribution with more good than bad individuals seems common in an EA run, at least towards the end of the run, when the algorithm produces many solutions close to the optimum, and a few outliers. The results below are averaged over 100,000 such randomly sampled populations.

We compare the frequentist probability of a good generation $\mathrm{PGG}_{\mathrm{iz},\delta^*}$ depending on the expected number of evaluations used by different procedures. To calculate $\mathrm{PGG}_{\mathrm{iz},\delta^*}$, we run the sampling mechanism and look at the resulting order according to sample means. If all decisions required by the scenario (i.e., those defined by $\mathcal{C}$) have been identified correctly taking into account the indifference zone, the run is counted as successful. Otherwise, it is not successful. $\mathrm{PGG}_{\mathrm{iz},\delta^*}$ is the percentage of correct runs. The parameters α^* and δ^* not only are determinants of $\mathrm{PGG}_{\mathrm{iz},\delta^*}$, but also of the expected total number of samples, $\mathrm{E}[N]$, for a given numerical experiment.

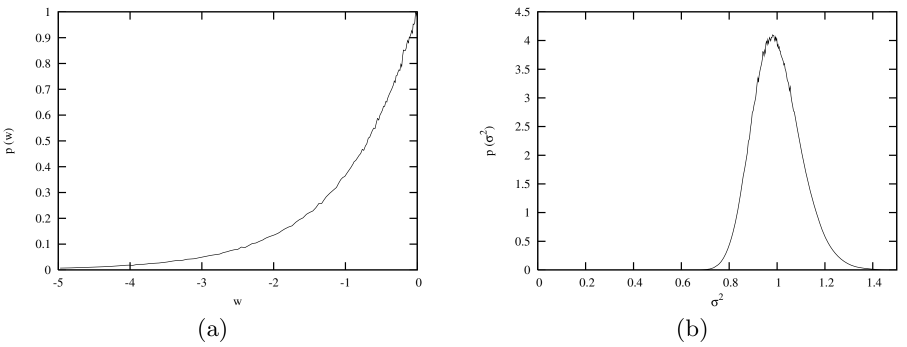


Fig. 2. Empirical distribution of means (a) and variances (b) for the empirical tests

The sampling mechanisms considered are:

1. A standard allocation scheme which samples all individuals equally often. This is denoted by Equal.
2. $\mathcal{OCBA}^{\text{EA}}_{\delta^*}$ for a steady-state EA with population size $\mu = 9$ and one offspring generated
3. $\mathcal{OCBA}^{\text{EA}}_{\delta^*}$ for an evolution strategy with $5, 10$ replacement.
4. $\mathcal{OCBA}_{\delta^*}$ to select the best of the 10 individuals.
5. $\mathcal{OCBA}_{\delta^*}$ to give a complete ranking of the 10 individuals.

For all tests, an indifference zone of $\delta^* = 0.2$ is used, i.e. the ordering of a pair of individuals is accepted as correct if the higher ranked individual is not

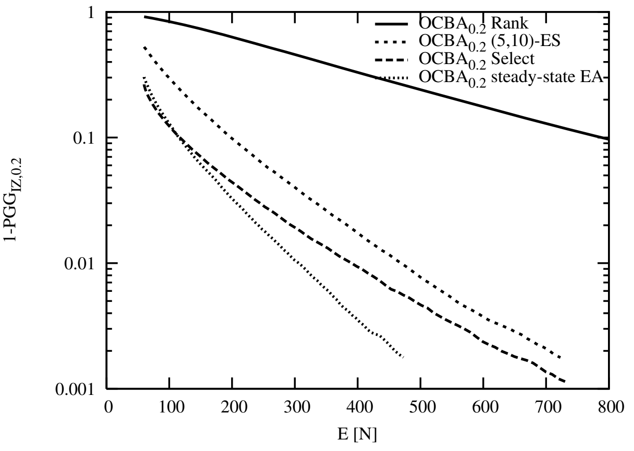


Fig. 3. Efficiency of different sampling mechanisms (expected number of samples required, E[N], to reach a certain level of $PGGiz$)

more than 0.2 worse than the lower ranked individual. We use the stopping rule $\mathrm{PGG}_{\mathrm{Slep},\delta^*} > 1-\alpha^*$, where α^* is varied to generate lines in Figure 3.

A complete ranking of the individuals is the most challenging task and requires the highest number of samples to reduce the error $1-\mathrm{PGG}_{\mathrm{iz},\delta^*}$. The curves for steady-state EA and (5,10) ES are significantly below the curve for the complete ranking, which shows that an EA indeed requires only partial information, and that a lot of samples can be saved by generating only the required information. Interestingly, the steady-state EA operation even requires less samples than identifying only the best individual ($\mathcal{OCBA}_{0.2}$Select). This is due to the fact that we generate the means according to a negative exponential distribution, i.e. there are several good but few bad individuals, and thus it is relatively easier to identify the worst individual and the better of two random pairs for the steady-state EA than it is to identify the best individual.

Figure 4 compares our new $\mathcal{OCBA}$-based EA with standard EAs using the same number of samples for each individual. The $\mathcal{OCBA}$-based sampling allocation schemes are much more efficient than the corresponding Equal allocation variants, which shows that integrating a statistical ranking and selection procedure is beneficial.

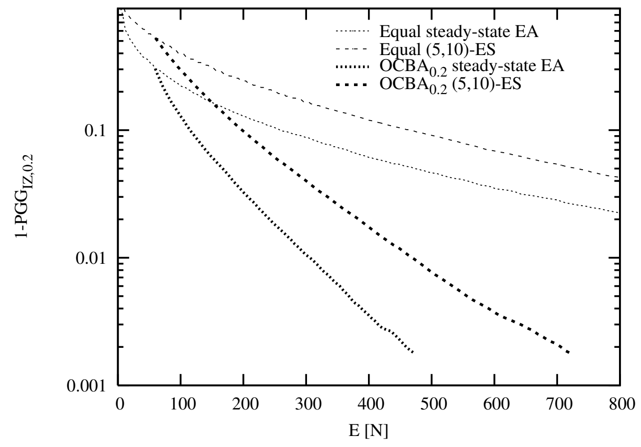


Fig. 4. Comparison of the new $\mathcal{OCBA}$-based EA (bold lines) and the standard EA with Equal sampling (thin lines)

For example, to reduce the probability of an erroneous generation, $1-\mathrm{PGG}_{\mathrm{iz},\delta^*}$, to 0.02, the standard $(5,10)$-ES would require an average of 1160 evaluations per generation. Our $\mathcal{OCBA}$-based ES achieves the same accuracy with 385 evaluations per generation. For the steady-state EA, the differences are even larger: the standard steady-state EA would require an average of 845 evaluations per generation. while our $\mathcal{OCBA}$-based EA only requires 240 evaluations. That is, our method saves 67-71% of the samples, and the benefit of our newly proposed methods increases with an increasing desired $\mathrm{PGG}_{\mathrm{iz},\delta^*}$.

While this only looks at a single example of an artificially generated iteration, it is to be expected that the benefit of our proposed method will be even larger for larger populations (because the required information becomes an even smaller portion of the information required for a full ranking) and in an iterative EA setting (because $\mathcal{OCBA}$ will be able to re-use the samples of surviving individuals, and those are the individuals that were allocated relatively many samples).

7 Conclusion

Optimization in noisy environments is challenging, because the noise makes it difficult to decide which of two solutions is better, which is a prerequisite for every optimization algorithm. While noise can usually be reduced by averaging over multiple samples, this is a costly process. In this paper, we have proposed a new adaptive sample allocation mechanism that attempts to minimize the number of samples required to warrant a proper functioning of an EA. The approach is based on two ideas:

1. Restriction of the focus on those pairwise comparisons that are actually used by the EA. As the empirical results have shown, these comparisons may require less samples than even only identifying the best individual.
2. The use of $\mathcal{OCBA}$, a sample allocation procedure from statistical ranking and selection. This allowed to distribute the additional evaluations to those individuals where they promise the highest benefit, and to stop sampling when there is sufficient evidence for correct selection.

More work is needed to identify the best way to take advantage of this proposal. As a next step, we will show the benefit of the proposed method not only on a hypothetical generation, but over a complete run of the EA. Then, the method will be extended to allow an efficient identification of the best individual encountered during the run. Finally, guidelines for setting the parameters α^* and δ^* as iterations proceed in order to obtain a better overall performance are needed.

References

1. D. V. Arnold and H.-G. Beyer. A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24:135–159, 2003.
2. T. Bartz-Beielstein, D. Blum, and J. Branke. Particle swarm optimization and sequential sampling in noisy environments. In *Metaheuristics International Conference*, 2005.
3. H.-G. Beyer. Toward a theory of evolution strategies: Some asymptotical results from the $(1 \overset{+}{,} \lambda)$-theory. *Evolutionary Computation*, 1(2):165–188, 1993.
4. J. Boesel. *Search and Selection for Large-Scale Stochastic Optimization.* PhD thesis, Northwestern University, Evanston, Illinois, USA, 1999.
5. J. Boesel, B. L. Nelson, and S. H. Kim. Usint ranking and selection to "clean up" after simulation optimization. *Operations Research*, 51:814–825, 2003.

6. J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.
7. J. Branke, S. Chick, and C. Schmidt. New developments in ranking and selection: An empirical comparison of the three main approaches. In N.E. Kuhl, M. N. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Winter Simulation Conference*, pages 708–717. IEEE, 2005.
8. J. Branke and C. Schmidt. Selection in the presence of noise. In E. Cantu-Paz, editor, *Genetic and Evolutionary Computation Conference*, volume 2723 of *LNCS*, pages 766–777. Springer, 2003.
9. J. Branke and C. Schmidt. Sequential sampling in noisy environments. In X. Yao et al., editor, *Parallel Problem Solving from Nature*, volume 3242 of *LNCS*, pages 202–211. Springer, 2004.
10. P. Buchholz and A. Thümmler. Enhancing evolutionary algorithms with statistical sselection procedures for simulation optimization. 2005.
11. E. Cantu-Paz. Adaptive sampling for noisy problems. In K. Deb et al., editors, *Genetic and Evolutionary Computation Conference*, volume 3102 of *LNCS*, pages 947–958. Springer, 2004.
12. C.-H. Chen. A lower bound for the correct subset-selection probability and its application to discrete event simulations. *IEEE Transactions on Automatic Control*, 41(8):1227–1231, 1996.
13. A. Di Pietro, L. While, and L. Barone. Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In *Congress on Evolutionary Computation*, pages 1254–1261. IEEE, 2004.
14. U. Hammel and T. Bäck. Evolution strategies on noisy functions, how to improve convergence properties. In Y. Davidor, H. P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, volume 866 of *LNCS*. Springer, 1994.
15. H. E. Hedlund and M. Mollaghasemi. A genetic algorithm and an indifference-zone ranking and selection framework for simulation optimization. In *Winter Simulation Conference*, pages 417–421. IEEE, 2001.
16. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.
17. P. Stagge. Averaging efficiently in the presence of noise. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature V*, volume 1498 of *LNCS*, pages 188–197. Springer, 1998.

The Role of Representations in Dynamic Knapsack Problems

Jürgen Branke[1], Merve Orbayı[2], and Şima Uyar[3]

[1] Institute AIFB, University of Karlsruhe, Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de
[2] Informatics Institute, Istanbul Technical University, Istanbul, Turkey
merve.orbayi@tikle.com
[3] Computer Engineering Department, Istanbul Technical University, Istanbul, Turkey
etaner@cs.itu.edu.tr

Abstract. The effect of different representations has been thoroughly analyzed for evolutionary algorithms in stationary environments. However, the role of representations in dynamic environments has been largely neglected so far. In this paper, we empirically compare and analyze three different representations on the basis of a dynamic multi-dimensional knapsack problem. Our results indicate that indirect representations are particularly suitable for the dynamic multi-dimensional knapsack problem, because they implicitly provide a heuristic adaptation mechanism that improves the current solutions after a change.

Keywords: Evolutionary algorithm, representation, dynamic environment.

1 Introduction

Many real-world problems are dynamic in nature. The interest in applying evolutionary algorithms (EAs) in dynamic environments has been increasing over the past years, which is reflected in the increasing number of papers on the topic. For an in-depth overview on the topic, see e.g. [2, 10, 12, 17].

Most of the literature attempts to modify the algorithm to allow a better tracking of the optimum over time, e.g. by increasing diversity after a change, maintaining diversity throughout the run, or incorporating a memory. In this paper, we focus on the representation's influence on an EA's performance in dynamic environments. Instead of searching the solution space directly, usually EAs search in a transformed space defined by the genetic encoding. This mapping between solution space and genetic search space is generally called "representation", or "genotype-phenotype mapping". The representation together with the genetic operators and the fitness function define the fitness landscape, and it is generally agreed upon that a proper choice of representation and operators is crucial for the success of an EA, see e.g. [15, 16].

Depending on the representation, the fitness landscape can change from being unimodal to being highly multimodal and complex, and thus the representation strongly influences the EA's ability to approach the optimum. In a dynamic environment, in addition to the (static) characteristics of the fitness landscape, the representation influences the characteristics of the fitness landscape dynamics, as has been recently demonstrated in [3]. Consequently, depending on the representation, the tracking of the optimum over time may be more or less difficult.

This paper examines the performance of three different genetic representations for the dynamic multi-dimensional knapsack problem (dMKP) [3]. The MKP is a well studied problem, and different representations have been proposed and compared e.g. in [6, 9, 14, 15]. For our study, the binary representation with a penalty for constraint handling is selected as an example of a direct representation. As indirect representations, we consider a permutation representation and a weight-coding. In the latter, the items' profits are modified and a simple deterministic heuristic is used to construct the solution. Intuitively, this last representation seems particularly promising for dynamic environments, as it naturally incorporates heuristic knowledge that would immediately improve a solution's phenotype after a change of the problem instance.

The paper is structured as follows: Section 2 introduces the dynamic MKP problem and briefly explains the different representations used in this study. The experimental results are reported and analyzed in Section 3. The paper concludes in Section 4 with a summary and some ideas for future work.

2 The Dynamic Multi-dimensional Knapsack Problem

Knapsack problems [11] are commonly used combinatorial benchmark problems to test the performance of EAs. The multi-dimensional knapsack problem (MKP) belongs to the class of NP-complete problems. The MKP has a wide range of real world applications such as cargo loading, selecting projects to fund, budget management, cutting stock, etc. It can be formalized as follows.

$$\text{maximize} \qquad \sum_{j=1}^{n} p_j \cdot x_j \tag{1}$$

$$\text{subject to} \sum_{j=1}^{n} r_{ij} \cdot x_j \leq c_i, \quad i = 1, 2, ..., m \tag{2}$$

where n is the number of items, m is the number of resources, $x_j \in \{0, 1\}$ shows whether item j is included in the subset or not, p_j shows the profit of item j, r_{ij} shows the resource consumption of item j for resource i and c_i is the capacity constraint of resource i.

For the MKP, several different genetic representations and genetic operators have been proposed. A detailed analysis and comparison for static environments can be found in [6] and more recently in [15]. In the following, we describe the representations selected for our study in dynamic environments.

2.1 Direct Representations for the MKP

A representation is called *direct* if it can be interpreted directly as a solution to the problem. For the MKP, this corresponds to an assignment of values to the variables x_i in the above definition, e.g. in the form of a bit string where each bit corresponds to an item, indicating whether an item should be included in the knapsack or not.

Binary Representation with Penalty. One drawback of direct representations is often the difficulty to maintain feasibility, as the search space contains many infeasible solutions. In our study, we use a simple penalty-based method to drive the search towards feasible regions of the search space. We apply the penalty mechanism recommended in [8], which guarantees that feasible solutions are always preferred over infeasible ones.

$$fitness(x) = f(x) - penalty(x) \tag{3}$$

$$penalty(x) = \frac{p_{max} + 1}{r_{min}} * max\{CV(x,i) \mid i = 1 \ldots m\} \tag{4}$$

$$p_{max} = max\{p_i \mid i = 1 \ldots m\} \tag{5}$$

$$r_{min} = min\{r_{ij} \mid i = 1 \ldots m, j = 1 \ldots n\} \tag{6}$$

$$CV(x,i) = max(0, \sum_{j=1}^{n} r_{ij} \cdot x_j - c_i) \tag{7}$$

where p_{max} is the largest profit value calculated as in Eq. 5, r_{min} is the minimum resource consumption calculated as in Eq. 6 and $CV(x,i)$ is the maximum constraint violation for the ith constraint c_i calculated as in Eq. 7. It should be noted that $r_{min} \neq 0$ must be ensured. As genetic operators bit flip mutation and uniform crossover are used.

2.2 Indirect Representations for the MKP

Indirect representations require to run a decoder to generate the solution based on the genotype. There are many possible indirect representations. Usually, a representation is preferred that decodes all elements of the search space into feasible solutions. Thus, it is not necessary to design complicated repair mechanisms or to use a penalty to ensure feasibility. In this paper, we look at the two indirect representations discussed below.

Permutation Representation. A popular indirect representation is the permutation representation [6, 7], where the search space consists of all possible permutations of the items. To obtain the phenotype (actual solution), a decoder starts with an empty set, then considers the items one at a time in the order specified by the permutation. If an item can be added without violating any constraint, it is included in the solution, otherwise not.

The decoder used by the permutation representation guarantees that only feasible solutions are generated. Furthermore, these solutions lie on the boundary of the feasible region in the sense that no additional items could be included without violating at least one constraint, which is a necessary condition for optimality. Thus, the decoder generates solutions that are of significantly higher quality than randomly generated solutions. In a dynamic environment, solutions are immediately "repaired" after a change such that they are again at the boundary of feasibility in the new environment.

In [9], a good setup for the permutation representation is recommended, including uniform order based crossover (UOBX) and insert mutation as variation operators. In insert mutation, a new position for an element is selected randomly. The mutated element is inserted into its new position and the other elements are re-positioned accordingly. In UOBX, some positions are transfered directly to the offspring from the first parent with probability $p_1 = 0.45$. Then, starting from the first position, undetermined positions are filled with missing items in the order of the second parent.

Real Valued Representation with Weight Coding. A more complex example for indirect representations is the weight-coding (WC) approach [14]. In the weight-coding technique, a candidate solution for the MKP consists of a vector of real-valued genes (biases) associated with each item. To obtain the corresponding phenotype, first the original problem P is transformed (biased) into a modified problem P' by multiplying the original profits of each item with the corresponding bias. Then, a fast heuristic is used to find a solution to P', and finally, the resulting solution (items to be placed in the knapsack) is evaluated based on the original problem. Raidl [14] discusses two possible decoding heuristics. The one using the surrogate relaxation method is preferred due to its lower computational requirements. The surrogate relaxation method [13] simplifies the original problem by transforming all constraints into a single one as follows:

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_i \cdot r_{ij} \right) x_j \leq \sum_{i=1}^{m} c_i \tag{8}$$

where a_i is the surrogate multiplier for the ith constraint, and r_{ij} is the resource coefficient.

Surrogate multipliers are determined by solving the relaxed MKP (i.e., variables x_i can take any value $\in [0, 1]$) by linear programming, and using the values of the dual variables as surrogate multipliers. Then, to obtain a heuristic solution to the MKP, the *profit/pseudo-resource consumption ratios* denoted as u_j are calculated as given in Eq. 9.

$$u_j = \frac{p_j}{\sum_{i=1}^{m} a_i r_{ij}} \tag{9}$$

The items are then sorted in decreasing order based on their u_j values, and this order is used to construct solutions just as for the permutation representation, i.e. items are considered one at a time, and if none of the constraints are violated, added to the solution. To keep computation costs low, in [14] the surrogate multiplier values a_i are determined only once for the original problem at the beginning. As a result, the decoding step starts with the computation of the u_j

values based on the biased profits. Note that in a dynamic environment, the WC representation requires to re-compute the pseudo-resource consumption values a_i after every change of the environment.

In [14], several biasing techniques are discussed and compared. We initialize the biases according to $w_j = 2^R$, where R is a uniformly generated variable in the range $[-1, 1]$ This leads to a distribution with many small and few larger values. For mutation, we deviate from the re-initialization of biases used in [14] and instead use Gaussian mutation with $\sigma = 1$. To generate the modified profits, the original profits are simply multiplied with the biases, i.e. $p'_j = p_j * w_j$. Uniform crossover is used as second genetic operator.

Since the permutation representation and the WC representation share similar construction mechanisms, they both benefit from the resulting heuristic bias. However, by calculating the pseudo-resource consumption values, the influence of heuristic knowledge for WC is even larger.

In dynamic environments, the WC representation appears to be particularly advantageous, for two reasons:

1. Because of the integration of heuristic knowledge, good solutions are generated right from the beginning, i.e., the algorithm improves more quickly. In dynamic environments, time is scarce (otherwise one could just regard the problem as a sequence of stationary problems), and the heuristic bias gives this representation a head start.
2. Changes of the environment are immediately taken into account by the underlying heuristic, which means that the existing solutions are heuristically adjusted after a change of the problem instance.

2.3 The Dynamic MKP

In our study, we use a dynamic version of the MKP as proposed in [3] and described below. Basis is the first instance given in the file *mknapcb4.txt* which can be downloaded from [1]. It has 100 items, 10 knapsacks and a tightness ratio of 0.25. For every change, the profits, resource consumptions and the constraints are multiplied by a normally distributed random variable as follows:

$$\begin{aligned} p_j &\leftarrow p_j * (1 + N(0, \sigma_p)) \\ r_{ij} &\leftarrow r_{ij} * (1 + N(0, \sigma_r)) \\ c_i &\leftarrow c_i * (1 + N(0, \sigma_c)) \end{aligned} \tag{10}$$

Unless specified otherwise, the standard deviation of the normally distributed random variable used for the changes has been set to $\sigma_p = \sigma_r = \sigma_c = 0.05$ which requires on average 11 out of the 100 possible items to be added or removed from one optimal solution to the next. Each profit p_j, resource consumption r_{ij} and constraint c_i is restricted to an interval as determined in Eq. 11.

$$\begin{aligned} lb_p * p_j \leq p_j \leq ub_p * p_j \\ lb_r * r_{ij} \leq r_{ij} \leq ub_p * r_{ij} \\ lb_c * c_i \leq c_i \leq ub_p * c_i \end{aligned} \tag{11}$$

where $lb_p = lb_r = lb_c = 0.8$ and $ub_p = ub_r = ub_c = 1.2$. If any of the changes causes any of the lower or upper bounds to be exceeded, the value is bounced back from the bounds and set to a corresponding value within the allowed boundaries.

3 Empirical Results

For this study, we used a more or less standard steady-state EA with a population size of 100, binary tournament selection, crossover probability of 1.0, and mutation probability of 0.01 per gene. The genetic operators crossover and mutation depend on the representation and have been implemented as described above. The new child replaces the current worst individual in the population if its fitness is better than the worst. The EA uses phenotypic duplicate avoidance, i.e. a child is re-generated if a phenotypically identical individual already exists in the population. This feature seems important in particular for indirect representations with high redundancy, i.e. where many genotypes are decoded to the same phenotype.

Unless stated otherwise, after a change, the whole population is re-evaluated before the algorithm is presumed. The genotypes are kept unless the change creates phenotypically identical individuals, in which case duplicates are randomized. As a performance measure, we use the error to the optimum. We use *glpk* [5] for calculating the surrogate multipliers for the WC and CPLEX for calculating the true optimum for each environment. All results are averages over 50 runs with different random seeds but on the same series of environment changes.

Note that the following analysis assumes that the evaluation is by far the most time-consuming operation (as is usual for many practical optimization problems), allowing us to ignore the computational overhead caused by the decoders.

3.1 Relative Performance in Stationary Environments

Figure 1 compares the three representations on a stationary environment, which will serve as a baseline for the comparison in dynamic environments. As can be seen, the permutation representation is fastest to converge, WC is somewhat slower but then takes over, and the binary representation with penalty is very slow, and remains worst throughout the run. The first (random) solution generated by the WC, permutation, and binary approaches has an error of approximately 6366, 7381, and 16374922, respectively. This shows that the WC representation has a higher heuristic bias than the permutation representation, while the binary approach starts with infeasible (more infeasibles with lower tightness ratios) and highly penalized solutions.

3.2 Dynamic Environment

The relative performance of the different representations in a dynamic environment is shown in Figure 2. In the plot, the fitness of the first individual after a

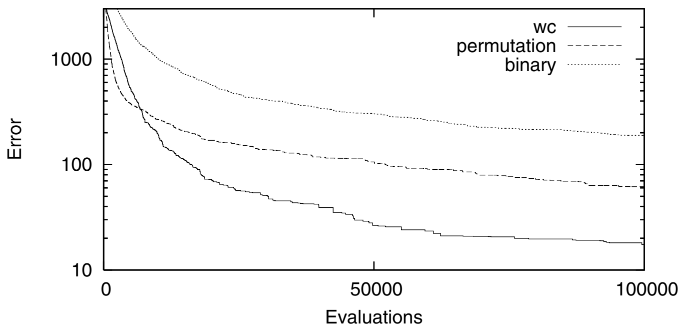


Fig. 1. Error over time for different representations in a stationary environment

change is indicated with (x) for the permutation approach and (o) for the WC approach. For further details see also Table 1.

Several interesting observations can be made. First, as expected, there is a significant increase in error right after a change. Nevertheless, the error after a change is much smaller than the error at the beginning of the run. Compared to the first environment, the average error of the starting solution in environments 2-10 is reduced by approximately 75% for WC, 71% for permutation and 96% for the binary representation with penalty. This means that all representations benefit dramatically from transferring solutions from one environment to the next. WC starts better than the permutation representation, and both indirect

Table 1. Average error or initial solution, the solution right before change, and the solution right after a change, ± standard error

	WC	permutation	binary
initial solution	6307±133	7471±140	15764226±418421
before change	197±10	260±15	834±33
after change	1482±67	2201±94	577226±47984

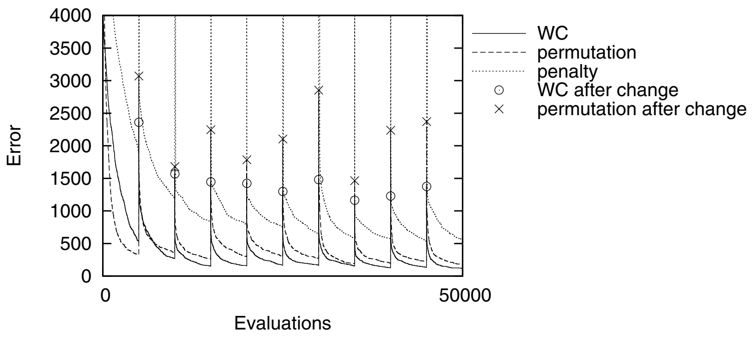


Fig. 2. Error over time for different representations in a dynamic environment

representations are much better than the binary one. The binary representation with penalty can not prevent the solutions to become infeasible, but recovers quickly. It clearly benefits most from re-using information, and it can improve its performance over several environmental periods (not only from the first to the second environment).

Second, starting from better solutions, the algorithms are able to find better solutions throughout the stationary periods. The benefit seems highest for the binary representation, while the permutation approach can improve performance only a little bit. At the end of the 10th environmental period (evaluation 50,000), the solution quality reached by the indirect representations is close to the error found after 50,000 evaluations in a stationary environment. This means that the algorithms don't get stuck at a previously good but now inferior solution.

Third, as in the stationary case, the WC representation outperforms the permutation representation after a while and performs best overall.

3.3 Restart

Instead of continuing the EA run after a change, one might also re-start the EA with a new, randomized population, which is a common strategy to avoid premature convergence of the population. In this case, improving the solution quality fast would be even more important. As Figure 3 shows, re-initializing the population after a change is worse than simply continuing for all three rep-

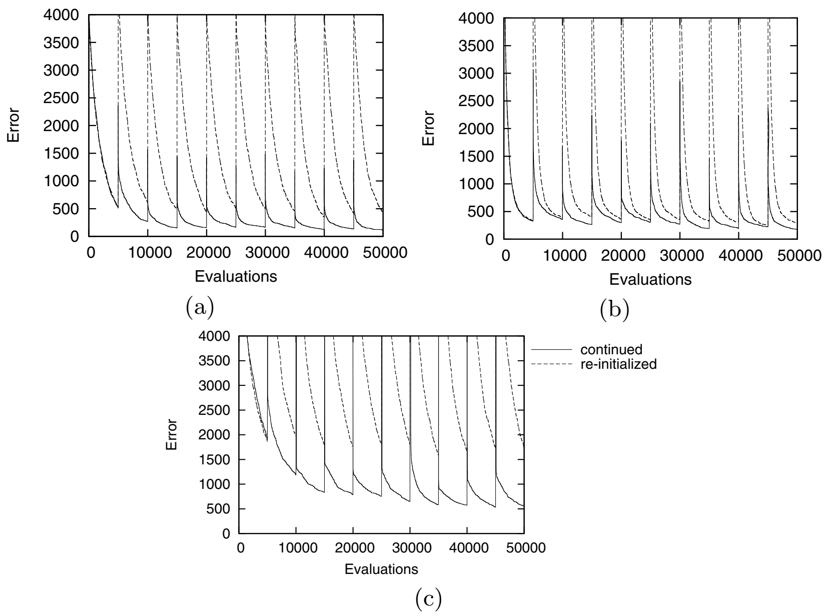


Fig. 3. Error over time for different representations in a dynamic environment. Comparison of keeping the population (solid line) or re-start (dashed line) after a change for the (a) WC (b) permutation and (c) binary representation.

resentations. For the permutation representation, the difference is the smallest, for binary representation it is largest. The results suggest that premature convergence is not so much an issue in the settings considered, either because the duplicate avoidance used is sufficient to prevent premature convergence, or because the population does not converge within the 5000 evaluations per period anyway.

3.4 Hypermutation

Hypermutation [4] has been suggested as a compromise between complete restart and simply continuing evolution. With hypermutation, the mutation probability is increased for a few iterations after a change to re-introduce diversity. For our experiments, we tried to increase mutation in such a way that it would have similar effects for all representations. For the binary representation, we increased the probability of mutation to $p_m = 0.05$ per gene. For WC, we increased the standard deviation for the Gaussian mutation to $\sigma = 5$. For the permutation representation, we applied insert mutation 5 times to each newly generated individual. In our tests, hypermutation had little effect except if the WC representation is used (not shown). Only for WC, hypermutation helped to speed up fitness convergence significantly in the first period, which indicates that either the mutation step size or the area for initialization have been chosen too small in the first environment.

3.5 Higher Change Frequency

Obviously, the higher the change frequency, the more important it is to produce good solutions quickly, and thus the higher should be the advantage of indirect representations. Figure 4 shows the same performance plots as in the previous subsection, but with a change every 2000 evaluations.

With the higher change frequency, the WC representation improves over the permutation representation only in the third environmental period, although according to the number of evaluations, the switch is actually earlier than in

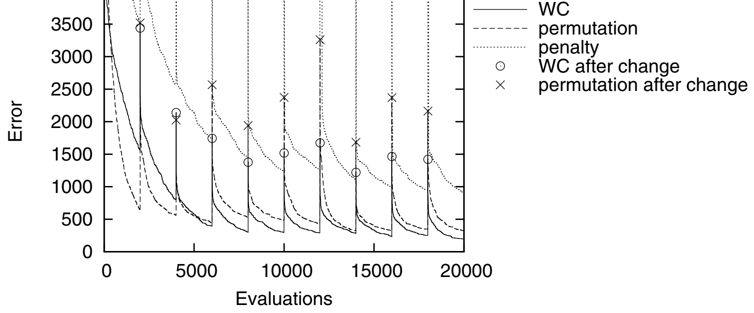


Fig. 4. Error over time for different representations in a dynamic environment with a change every 2000 evaluations

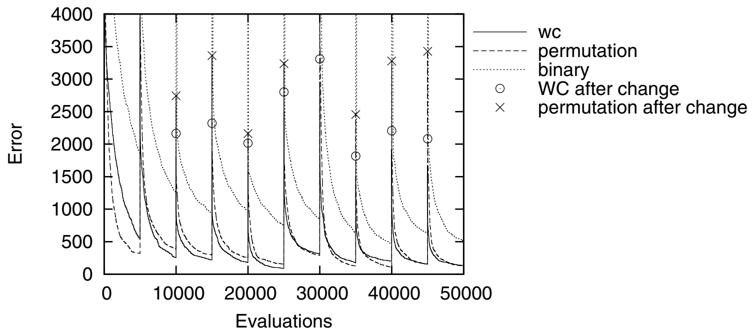


Fig. 5. Error over time for different representations in a highly severe dynamic environment

the previous case with lower change frequency. The binary representation keeps improving over all 10 environmental periods.

3.6 Effect of Change Severity

The aim of this experiment is to see the effect of change severity. To implement a more severe change, in Eq. 10, we set the standard deviation of the normally distributed random variable used for the changes to $\sigma_p = \sigma_r = \sigma_c = 0.1$ and each profit p_j, resource consumption r_{ij} and constraint c_i is restricted to an interval as determined in Eq. 11 with $lb_p = lb_r = lb_c = 0.5$ and $ub_p = ub_r = ub_c = 1.5$.

The results for this more severe environment, shown in Figure 5 look very similar to the standard case we have looked at in the above subsections. The error immediately after a change is significantly higher, but all algorithms adapt rather quickly and the solutions found later on are comparable to those in the standard case. As the analysis of the offline error below will show, in particular the binary encoding suffers from the increased severity. The indirect encodings are barely affected.

4 Discussion and Conclusion

The results have demonstrated that the representation can have a tremendous effect on an EA's performance in dynamic environments. While the permutation representation was fastest to converge in terms of solution quality, the WC representation was best in coping with the dynamics of the problem. The binary representation with penalty performed extremely poor, as it improves slowly and is not even able to maintain feasibility after a change.

In a stationary environment, what usually counts is the best solution found at the end. After 20000 evaluations, the obtained errors of the approaches are 73 for WC, 170 for permutation, and 570 for binary representation with penalty. However, in a dynamic environment usually the optimization quality over time is important. Table 2 summarizes all the results by looking at the offline error, i.e. the average error over evaluations 5000-20000. This interval has been chosen because it was covered in all experiments, and removes the influence from the initial "warm-up" phase.

In the stationary case, WC representation performs best, with permutation a close second (+39%), and binary representation with more than four times the offline error of the two indirect representations. In a dynamic environment, when the algorithm is restarted after every change, the permutation representation benefits from its fast fitness convergence properties and performs best, while the binary representation improves so slowly and generates so many infeasible solutions that it is basically unusable. If the algorithms are allowed to keep the individuals after a change, they all work much better than restart. With increasing change frequency or severity, the performance of all approaches suffers somewhat, but the gap between the best-performing WC representation and the direct binary representation increases from 532% in the dynamic baseline scenario to 867% in the high frequency scenario and 3233% in the high severity scenario.

Table 2. Offline error of different representations in different environments Evaluations 5000-20000

	WC	permutation	binary
stationary	179.1	248.1	947.8
restart	1581.6	1115.2	625746.0
dynamic base	342.2	470.4	1823.0
high frequency	397.1	561.5	3445.7
high severity	456.4	621.6	14756.9

Overall, our results indicate that indirect representations, and in particular those with a strong heuristic component like weight-coding, may have clear advantages in dynamic environments, in addition to the advantages they provide in stationary environments. As future work, we will verify this hypothesis also for the travelling salesman problem and look at representations with explicit repair mechanisms.

Acknowledgements

We would like to thank Erdem Salihoglu and Michael Stein for their help regarding glpk and CPLEX.

References

1. J. E. Beasley. Or-library. online, http://www.brunel.ac.uk/depts/ma/research/jeb/orlib/mknapinfo.html.
2. J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer, 2001.

3. J. Branke, E. Salihoglu, and S. Uyar. Towards an analysis of dynamic environments. In *Genetic and Evolutionary Computation Conference*, pages 1433–1439. ACM, 2005.
4. H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA, 1990.
5. GLPK. GNU linear programming kit. online, http://www.gnu.org/software/glpk/glpk.html.
6. J. Gottlieb. *Evolutionary Algorithms for Combinatorial Optimization Problems.* Phd, Technical University Clausthal, Germany, December 1999.
7. J. Gottlieb. Permutation-based evolutionary algorithms for multidimensional knapsack problems. In *ACM Symposium on Applied Computing*, volume 1, pages 408–414. ACM, 2000.
8. J. Gottlieb. On the feasibility problem of penalty-based evolutionary algorithms for knapsack problems. In E.J.W. Boers et al., editors, *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pages 50–60. Springer, 2001.
9. G. R. Raidl J. Gottlieb. The effects of locality on the dynamics of decoder-based evolutionary search. In *Genetic and Evolutionary Computation Conference*, pages 283–290. Morgan Kauffman, 2000.
10. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.
11. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.
12. R. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments.* Springer, 2004.
13. H. Pirkul. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics*, 34:161–172, 1987.
14. G. R. Raidl. Weight-codings in a genetic algorithm for the multiconstraint knapsack problem. In *Congress on Evolutionary Computation*, pages 596–603. IEEE, 1999.
15. G. R. Raidl and J. Gottlieb. Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation*, 13(4), 2005.
16. F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms.* Physica, 2002.
17. K. Weicker. *Evolutionary Algorithms and Dynamic Optimization Problems.* Der Andere Verlag, 2003.

The Effect of Building Block Construction on the Behavior of the GA in Dynamic Environments: A Case Study Using the Shaky Ladder Hyperplane-Defined Functions

William Rand[1] and Rick Riolo[2]

[1] Northwestern University, Northwestern Institute on Complex Systems,
Evanston, IL, 60208-4057, USA
wrand@northwestern.edu

[2] University of Michigan, Center for the Study of Complex Systems,
Ann Arbor, MI 48109-1120, USA
rlriolo@umich.edu

Abstract. The shaky ladder hyperplane-defined functions (sl-hdf's) are a test suite utilized for exploring the behavior of the genetic algorithm (GA) in dynamic environments. We present three ways of constructing the sl-hdf's by manipulating the way building blocks are constructed, combined, and changed. We examine the effect of the length of elementary building blocks used to create higher building blocks, and the way in which those building blocks are combined. We show that the effects of building block construction on the behavior of the GA are complex. Our results suggest that construction routines which increase the roughness of the changes in the environment allow the GA to perform better by preventing premature convergence. Moreover, short length elementary building blocks permit early rapid progress.

1 Introduction

In order to conduct controlled observations on the GA in dynamic environments, a test suite of problems is necessary, so that we can control the inputs to the system and define metrics for the outputs. Moreover, the more parameters of the system (e.g. time and severity of shakes, difficulty of the problem) that are controllable, the easier it is to test explanations for the observed behavior.

Other test suites for EAs in dynamic environments exist, such as the dynamic knapsack problem, the moving peaks problem and more [1]. The test suite and its variants presented here are similar to the dynamic bit matching functions utilized by Stanhope and Daida [2] among others. The test functions that we developed to explore the GA in dynamic environments, the shaky ladder hyperplane-defined functions (sl-hdf's) [3], are a subset of the hdf's [4]. Holland created these functions in part to meet criteria developed by Whitley [5]. The hdf's, designed to represent the way the GA searches by combining building blocks, are well-suited for understanding the operation of the GA [6] [7] [8]. Extending our previous work on sl-hdf's [6] [7] [8], here we explore the effect of different types of building blocks on the GA by manipulating the way in which intermediate building blocks within the sl-hdf's are constructed. Moreover, whereas in past papers we often examined the effect of the severity of change within a single environment type, here we compare the effect of different environment types while holding the severity of change relatively constant.

We begin by describing the sl-hdf's and three variants. We then describe the experiments with these functions, examine the behavior of the GA, discuss the results and draw some conclusions.

2 Shaky Ladder Hyperplane-Defined Functions and the Variants

In this section we describe the sl-hdf's and the three variants we will be exploring. For an in-depth explanation of the construction of the sl-hdf's see [8].

The sl-hdf's are a subset of Holland's hdf's [4]. To make the hdf's usable as a test platform in dynamic environments we place three restrictions on the hdf's: (1) The *Unique Position Condition* (UPC), which requires that all elementary schemata contain no conflicting bits; (2) The *Unified Solution Condition* (USC), which guarantees that all of the specified bits in the positive-valued elementary level schemata must be present in the highest level schema, and that all intermediate schema are a composition of lower level schema; and (3) The *Limited Pothole Cost Condition* (LPCC), which states that the fitness contribution of any pothole plus the sum of the fitness contributions of all the building blocks in conflict with that pothole must be greater than zero.

These three conditions guarantee that any string that matches the highest level schema must be optimally valued. Moreover it gives us an easy way to create a similar but different sl-hdf by changing the intermediate building blocks. This process is referred to as "shaking the ladder" [3], i.e. the intermediate schemata are changed which alters the reward structure that describes the fitness landscape. Thus these restrictions allow us to transform the full class of hdf's into a class that can be used for exploring the behavior of the GA on dynamic environments.

There are many parameters that control the construction of the sl-hdf's. Next we explain and explore the parameters that affect the three variant sl-hdf construction methods used in this paper.

2.1 Elementary Schemata Length

The elementary schemata length (l) is the distance between fixed bits in the elementary schemata. For the first two variants described below (Cliffs and Smooth), the elementary schemata length is not specified. When the elementary schemata length is not specified the fixed bit locations are chosen randomly from the whole string. On average when the length is unspecified, the actual length will be large, nearing the length of the string. For the third variant (Weight), the elementary schemata length is set to $l = \frac{l_s}{10} = 50$, where l_s is the overall length of the string. This relationship is taken from Holland [4].

2.2 Mean and Variance of Schemata Weight

To create the schemata we need to specify the weight (w) that each schema contributes to the overall fitness function. Two parameters, mean and variance of the schemata weight, are used to specify the normal distribution from which the weight for each intermediate schemata is drawn. There is one caveat to this: since a normal distribution would allow negative weights to be drawn, and since the proof of the optimality of the highest level schemata requires that the intermediates always increase the overall value of the schemata, a weight is redrawn from the same distribution if its value is not positive. In all of the experiments described herein, the weight of the elementary schemata (2), potholes (-1) and highest level schemata (3) remains unchanged. In the first two variants (Cliffs and Smooth), the weight of the intermediate schemata is also held constant at 3. However in the third variant (Weight) the weight of each intermediate schemata is drawn from a distribution with mean of 3, and variance of 1.

2.3 Restricted vs. Unrestricted Intermediate Construction

In the sl-hdf there are three groups of schemata held constant, the elementary schemata, the potholes, and the highest level schema. The fourth set of schemata, the intermediate schemata, is the group of schemata that changes. Thus the intermediate schemata could either be constructed out of any of the fixed groups of schemata, which is called the unrestricted construction method, or the intermediate schemata could be constructed out of just the elementary schemata, which is called the restricted construction method and is more similar to Holland's original description [4]. The unrestricted construction method is used in the first variant (Cliffs) below, while the restricted method is used in the other two variants (Smooth and Weight).

2.4 Random vs. Neighbor Intermediate Construction

The schemata utilized for construction of the next level of intermediate schemata are selected randomly or from a prescribed order from the previous level. The random construction method creates new intermediate schemata at level n, by randomly choosing without replacement two schemata from level $n-1$ and combining them. In the neighbor construction method, all of the schemata at level $n-1$ are sorted by the location of their *centers*, which is the index value half the distance between their left and right fixed loci. The first two schemata that are next to each other in this sorted list are combined, and then the next two, until all pairs have been combined.

If the random construction routine has been used then when the ladder shakes, all of the intermediate schemata are destroyed and new ones are created by randomly combining the lower level schemata to create the intermediate schemata, and weights are assigned by drawing from the distribution specified by the intermediate weight mean and variance. If the neighbor construction routine is specified, then the intermediate schemata are left alone and instead new weights are drawn for the intermediate schemata. Thus when the neighbor construction routine is used the only thing that changes during the shakes of the ladder are the weights, and therefore this is sometimes called "shaking by weight." When the random construction routine is used then the whole form of the ladder changes and thus this is sometimes called "shaking by form."

2.5 The w_δ Parameter

w_δ is the fraction of intermediate schemata weights that are changed when the ladder is shaken. This parameter only makes sense with the neighbor intermediate schemata construction method, since in the random method all weights are changed when the ladder is shaken. However in the first two variants described below (Cliffs and Smooth), the variance of the weights is 0, thus the weights are all the same. In the last variant (Weight) $w_\delta = 1$ and the neighbor construction method is utilized, which results in shaking all the weights every time.

3 Variations on the sl-hdf

Given all of the parameters described above, it is necessary to determine how to explore this space systematically. One of the major choices is to decide whether to utilize the restricted or unrestricted intermediate construction technique. Initially we utilized the unrestricted technique, because it increased the amount of variety in the landscape. One of the properties of this technique is that it creates a large pool of material to draw from for the creation of the intermediate schemata. Thus once this decision has been made it is logical to use the shaking by form technique to sample widely from this pool of building blocks. This creates the most diverse and unrestrained set of building blocks. This unrestricted, shaking by form landscape became the *Cliffs* variant.

This variant does indeed produce interesting results, but we wanted to create something more similar to the original hdf's [4] and the best way to do that was to restrict the intermediate construction method. This small restriction changes the composition of the building blocks in a minor way, but does nothing to change the fact that they are dramatically changed every time the ladder shakes. This restricted, shaking by form landscape became the *Smooth* variant.

Finally, in order to move even closer to Holland's original description we utilized the neighbor intermediate construction technique. This technique dramatically alters the way intermediate building blocks are constructed in this test suite. However, since this technique fixes the particular schemata that are combined it is necessary to use a new way to introduce dynamics into the landscape. The only property a schema has besides its particular form is the weight assigned to it. Thus it is clear that if the form of the schema is held constant, in order to create dynamics the weights must be changed. Moreover, in order to make this variant as close to Holland's as possible we also restricted the length of the elementary building blocks. This restricted, shaking by weight landscape became the *Weight* variant.

The main differences between these variants as described in the preceding paragraphs are detailed in Table 1. We will describe each of these variants in turn in the following sections.

3.1 Cliffs Variant: Intermediate Schemata Are Cliffs

Figure 1 illustrates the Cliffs landscape, the base case in this paper.

The major differences between this variant and the other two are that it uses the unrestricted construction method. When creating a new intermediate schemata using the unrestricted method, all of the previous level schemata, plus the potholes, and the highest level schemata can be used to generate the new schemata. This has the property of introducing "cliffs" into the landscape, because the combination of any schemata and the highest level schemata is the highest level schemata. Thus many intermediate schemata are replaced by copies of the highest level schemata. An effect of this is that any string which matches the highest level schema will have a much higher value relative to the other strings than it would in the restricted construction method. Moreover, the effect of having some intermediate schemata combining with potholes or potholes combining with potholes to create intermediate schemata is interesting and complicated. Essentially, this has the effect of smoothing some transitions since some of the potholes will not be as detrimental as they could be. On the other hand, if there is one of these "bridges" in place and the ladder shakes removing the bridge, an individual making use of that bridge will suffer a sharp decline in fitness because, it loses an intermediate schemata and gains a pothole. In general the result of all these effects is that the fitness landscape is more sharply defined.

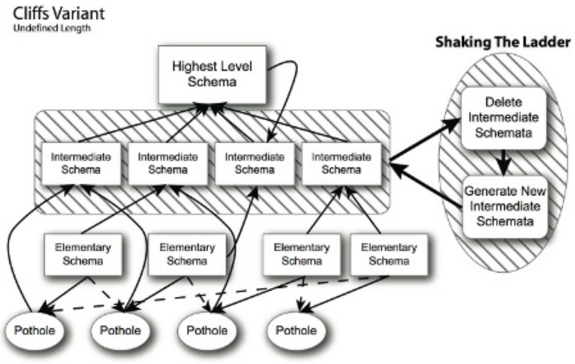


Fig. 1. Illustration of Shaky Ladder Construction and the Process of Shaking the Ladder for the Cliffs Variant

As to the rest of the parameters, the length of the schemata is not specified. The order of the schemata is set to 8. The length of the string is 500. The number of elementary schemata is 50. The mean of the intermediate schemata weight is 3 with a variance of 0. The construction method for the intermediate schemata is unrestricted and random.

3.2 Smooth Variant: An sl-hdf Without Cliffs

The variant that we refer to as the Smooth sl-hdf throughout this paper is constructed using the restricted method, which is the only difference between it and the Cliffs variant, and can be seen in Figure 2.

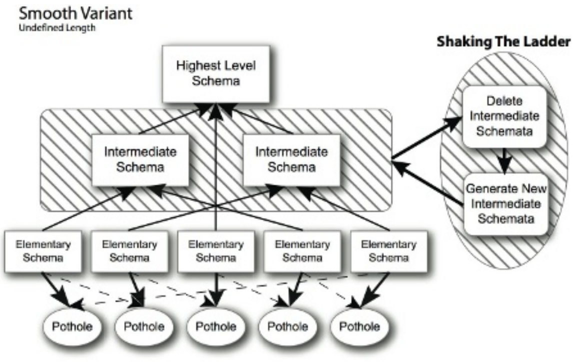


Fig. 2. Illustration of Shaky Ladder Construction and the Process of Shaking the Ladder for the Smooth Variant

This variant is called the Smooth sl-hdf because unlike the sl-hdf with Cliffs, there are no sharp edges in the landscape. Instead elementary schemata are combined to form intermediate schemata, which then are combined to form the next level of intermediate schemata and so forth.

To fully specify the parameters, the length of the schemata is not specified. The order of the schemata is set to 8. The length of the string is 500. The number of elementary schemata is 50. The mean of the intermediate schemata weight is 3 with a variance of 0. The construction method for intermediate schemata is restricted and random.

3.3 Weight Variant: Shaking the Weights Not the Form

This variant most closely resembles the hdf's described by Holland. It has limited length building blocks, and a highly restricted intermediate building block construction routine, as is explained in Figure 3.

To fully specify the parameters, the length of the schemata 50. The order of the schemata is set to 8. The length of the string is 500. The number of

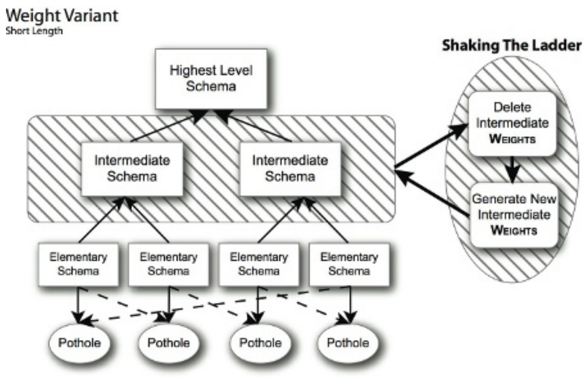


Fig. 3. Illustration of Shaky Ladder Construction and the Process of Shaking the Ladder for the Weight Variant

elementary schemata is 50. The mean of the intermediate schemata weight is 3 with a variance of 1. The construction method for intermediate schemata is restricted, as mentioned, and neighbor.

4 The Experiments and Results

The basic parameters for the experiment are laid out in Table 1.

Table 1. Initial GA and sl-hdf Parameters

Parameter	Cliffs Variant	Smooth Variant	Weight Variant
Population Size	1000		
Mutation Rate	0.001		
Crossover Rate	0.7		
Generations	1800		
String Length	500		
Selection Type	Tournament, size 3		
Number of Elem. Schemata	50		
Elementary Schemata Order	8		
Elementary Schemata Length	Not Specified		50
Mean, Var. of Int. Schem. Wt.	3, 0		3, 1
Int. Constr. Method	Unrestricted, Random	Restricted, Random	Restricted, Neighbor
t_δ	1, 25, 100, 900, 1801		
w_δ	0		1
Number of Runs	30		

4.1 Cliffs Variant Results

To more closely compare the three behavioral regimes of interest, we examined three values of t_δ: 1, 100 and 1801 for whole runs [1]. To provide some description of

[1] There are quantitative differences between t_δ values other than those presented here. We concentrate on these values for illustrative purposes.

the distribution (and variance) of the results, Figure 4 illustrates both the fitness of the best individual in the population (Best Performance) and the average fitness of the population (Average Performance) for the three t_δ values, averaged across 30 runs, presented every tenth generation.

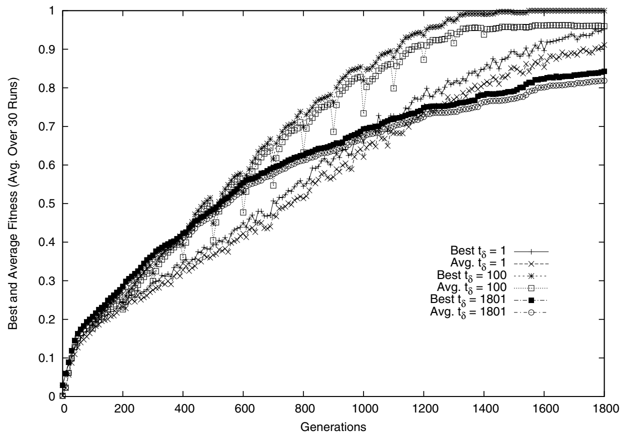


Fig. 4. Cliffs Variant: Performance Results of $t_\delta = 1801, 100, 1$

These results show that the regularly changing environment is able to outperform the static environment in the long run. Initially the dynamic environment under-performs the static environment but, before halfway through the run, the dynamic environment has achieved a superior fitness in both the best individuals and the average fitness of the populations. This is because the regularly changing environment prevents the GA from being locked into particular building blocks and forces it to explore a wider range of schemata. In the future, we hope to substantiate this hypothesis. For instance, the particular schemata located in each individual in the population could be examined.

In addition, Figure 4 shows that the Best and Average Performance results for the constantly changing environment surpass the static environment. It also is interesting to note that the gap between the performance of the best and average individuals in the constantly changing environments appears to be larger than it is in the other two environments. This is due to the fact that the constant dynamism of the $t_\delta = 1$ environment means that more individuals receive a 0 score in every generation than in the other two environments, thus driving down the Average Performance of the system relative to the Best Performance.

In a similar way, when the ladder is shaken in the regularly changing environment, the Average Performance of the system falls farther than the Best Performance of the system. This makes sense–when the ladder is shaken many of the individuals that were being rewarded before lose those rewards and hence their fitness falls greatly; however it is reasonable to suppose that there are

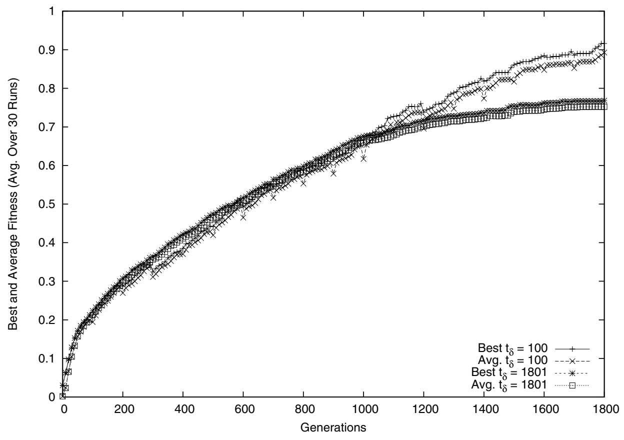


Fig. 5. Smooth Variant: Performance Results of $t_\delta = 1801, 100, 1$

some individuals (the new best individuals) immediately after a shake that have a higher performance than they did before the shake and thus they mitigate the fall of Best Performance.

4.2 Smooth Variant Results

In Figure 5, the results have been pared down to show the Best and Average Performance in the changing and static regimes[2]. Again, it is clear that a GA in a regularly changing environment is able to outperform a GA in a static environment in the long run. In this variant, the changing environment outperforms the static environment earlier on, but the GA does not perform as well as in the Cliffs environment. This is because the landscape changes in this variant are not rough enough to prevent the premature convergence [8].

In addition, like in the Cliffs variant, the Average Performance of the system falls farther than the Best Performance during a shake. This effect is due to the fact that individuals that were not the best in the past become the best after a shake, mitigating the decrease even though the population on average loses more fitness. Moreover it is interesting to note that the performance hits at these shakes do not appear to be as dramatic as they are in the Cliffs variant, for either the Average or Best Performance. This is one piece of evidence that shakes are not as violent in the Smooth variant as they are in the Cliffs variant.

4.3 Weight Variant Results

In Figure 6, the results again have been pared down to show the Best and Average Performance of the GA in the changing and static regimes, and as

[2] $t_\delta = 1$ is not presented since its behavior is qualitatively similar to $t_\delta = 100$.

in the Smooth variant the constantly changing environment is omitted. This graph is very different from the previous two graphs that examined the Best and Average Performance of the other two variants. Clearly there is virtually no difference between the static and changing environments.

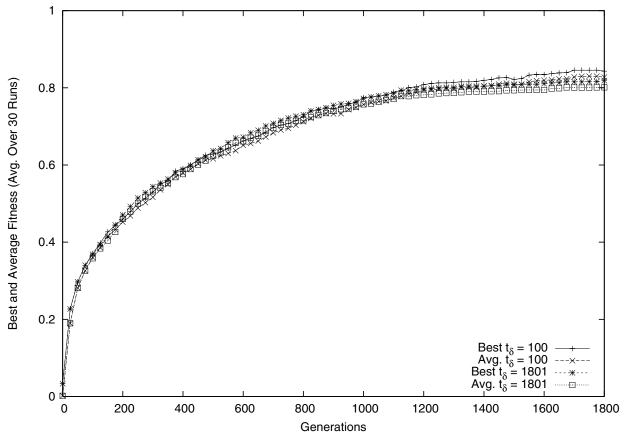


Fig. 6. Weight Variant: Performance Results of $t_\delta = 1801, 100$

In addition, the GA operating in this variant underperforms the GA's operating in the Smooth and Cliffs variants. This is because as the severity of the shakes decreases the GA is more likely to converge on suboptimal solutions [8]. On the other hand, in this variant the GA is able to make much more rapid progress early on, because the short elementary building blocks are easy to discover.

Moreover, the phenomenon that was observed before where the Average Performance falls farther than the Best Performance after a shake is not seen here. The loss of this effect is because the best individuals in the population are simply not that distinct from the average individuals. In addition, the loss due to a shake of the ladder is not that great and thus all individuals fall, but they fall a small amount. This, of course, is further confirmation that the shakes in this variant do very little to disrupt the GA's search process and thus this is the smoothest landscape of the three variants presented here.

In sum, across all three variants, we go from a situation where we have three distinct classes of phenomenon in the Cliffs landscape ("quasi-static", "regularly changing", "constantly changing"), to two distinct classes of phenomenon in the Smooth landscape ("quasi-static", "changing"), to one class of phenomenon in the Weights landscape. This leads to the hypothesis that bigger and different types of changes can produce different ranges of behavior in the GA. One way to confirm this hypothesis in the future would be to see if the GA exhibits similar behavior in other landscapes, or even in some of these landscapes (e.g. the Weight variant), when the way that the shake occurs is changed.

5 Conclusion and Future Work

This paper describes three different methods for constructing sl-hdf's, by varying the way basic building blocks are created and combined. GA behavior is dramatically different on the three variants, which suggests these (and other) variant sl-hdf's can be used to systematically explore GA behavior on wide range of dynamic environments. For example, we found that for the landscapes defined by some sl-hdf construction methods (the Cliffs and Smooth variants), a dynamic environment is actually preferable to a static environment because it prevents the GA from prematurely converging. In addition, the results indicate that increasing the severity of these shakes can increase the overall performance. Also, short elementary building blocks can lead to rapid progress in the early generations of a GA run. These observations may lend some guidance to application practitioners attempting to utilize GA's in dynamic environments. To be specific, one recommendation is to examine carefully the reward structure for the GA; by properly manipulating the reward structure, a practitioner can increase the efficacy of the GA. For example, in the job scheduling domain, a practitioner could alter the rewards received for scheduling combinations of jobs, which would be similar to changing the reward structure of the building blocks.

Two observations of the GA's behavior on the sl-hdf's studied here suggest properties of sl-hdf's that are associated with high GA performance in dynamic environments. First, we found that sl-hdf's with short elementary schemata lead to rapid performance increases early on relative to sl-hdf's built from longer schemata. Second, the Cliffs shaking method allows for an increase in performance because it prevents premature convergence. These two observations suggest other ways to construct sl-hdf's that should lead to even better GA performance. For instance, we defined a new sl-hdf variant, *Defined Cliffs*, in which we combine short elementary building blocks with the Cliffs shaking method. Preliminary results have shown that the Defined Cliffs variant has a superior performance to any of the results presented in this paper. A more in-depth discussion of these results will be presented in a future paper.

The results presented here show that different ways of constructing and combining building blocks in sl-hdf's can dramatically change the overall "texture" of the dynamic landscapes experienced by the GA. However, these results are only an initial examination of what effect different components of the building block construction process have on the performance of a GA. That is, because the interacting nature of some of these construction methods (how elementary building blocks are defined and combined) can create complex relationships between the construction algorithm and the overall texture of the landscape, it will take additional systematic studies to tease apart these relationships. In addition, the fact that the sl-hdf's also are dynamic, further complicates matters, and makes it difficult to determine if a landscape is easy or hard. Thus it is clear that future work exploring both the construction space of sl-hdf's and the dynamics that can be employed within this space is warranted to describe the overall effect of building blocks on the performance of the GA.

In conclusion, these experiments show that the construction of building blocks can dramatically change the behavior of the GA. By beginning to understand how the composition of building blocks can affect this behavior, practitioners can start to gain insight into how to better utilize GA's in real world problems.

Acknowledgments. We thank John Holland for continual insight into how to explore this space of functions. We acknowledge the contribution of Brooke Haueisen in generating the pictures of the sl-hdf's. The University of Michigan's Center for the Study of Complex Systems (CSCS) provided computational resources and support for RR.

References

1. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers (2001)
2. Stanhope, S.A., Daida, J.M.: Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment. In: Evolutionary Programming VII. Number 1447 in LNCS, Springer (1998) 693–702
3. Rand, W., Riolo, R.: Shaky ladders, hyperplane-defined functions and genetic algorithms: Systematic controlled observation in dynamic environments. In Rothlauf, F.e.a., ed.: Applications of Evolutionary Computing, Evoworkshops: EvoBIO, EvoCOMNET, EvoHot, EvoIASP, EvoMUSART, and EvoSTOC. Volume 3449 of Lecture Notes In Computer Science., Springer (2005)
4. Holland, J.H.: Building blocks, cohort genetic algorithms, and hyperplane-defined functions. Evolutionary Computation **8** (2000) 373–391
5. Whitley, D., Rana, S.B., Dzubera, J., Mathias, K.E.: Evaluating evolutionary algorithms. Artificial Intelligence **85** (1996) 245–276
6. Rand, W., Riolo, R.: The problem with a self-adaptive mutation rate in some environments: A case study using the shaky ladder hyperplane-defined functions. In Beyer, H.G.e.a., ed.: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005, New York, ACM Press (2005)
7. Rand, W., Riolo, R.: Measurements for understanding the behavior of the genetic algorithm in dynamic environments: A case study using the shaky ladder hyperplane-defined functions. In Beyer, H.G.e.a., ed.: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005, New York, ACM Press (2005)
8. Rand, W.: Controlled Observations of the Genetic Algorithm in a Changing Environment: Case Studies Using the Shaky Ladder Hyperplane-Defined Functions. PhD thesis, University of Michigan (2005)

Associative Memory Scheme for Genetic Algorithms in Dynamic Environments

Shengxiang Yang

Department of Computer Science, University of Leicester,
University Road, Leicester LE1 7RH, United Kingdom
s.yang@mcs.le.ac.uk

Abstract. In recent years dynamic optimization problems have attracted a growing interest from the community of genetic algorithms with several approaches developed to address these problems, of which the memory scheme is a major one. In this paper an associative memory scheme is proposed for genetic algorithms to enhance their performance in dynamic environments. In this memory scheme, the environmental information is also stored and associated with current best individual of the population in the memory. When the environment changes the stored environmental information that is associated with the best re-evaluated memory solution is extracted to create new individuals into the population. Based on a series of systematically constructed dynamic test environments, experiments are carried out to validate the proposed associative memory scheme. The environmental results show the efficiency of the associative memory scheme for genetic algorithms in dynamic environments.

1 Introduction

Genetic algorithms (GAs) have been applied to solve many optimization problems with promising results. Traditionally, the research and application of GAs have been focused on stationary problems. However, many real world optimization problems are actually dynamic optimization problems (DOPs) [4]. For DOPs, the fitness function, design variables, and/or environmental conditions may change over time due to many reasons. Hence, the aim of an optimization algorithm is now no longer to locate a stationary optimal solution but to track the moving optima with time. This challenges traditional GAs seriously since they cannot adapt well to the changing environment once converged.

In recent years, there has been a growing interest in investigating GAs for DOPs. Several approaches have been developed into GAs to address DOPs, such as diversity maintaining and increasing schemes [5, 7, 11], memory schemes [2, 14, 17], and multi-population approaches [3]. Among the approaches developed for GAs in dynamic environments, memory schemes have proved to be beneficial for many DOPs. Memory schemes work by storing useful information, either implicitly [6, 9, 12] or explicitly, from the current environment and reusing it later in new environments. In [17, 19], a memory scheme was proposed into population-based incremental learning (PBIL) [1] algorithms for DOPs, where the working probability vector is also stored and associated with the best sample it creates in the memory. When the environment changes, the stored probability vector can be reused in the new environment.

In this paper, the idea in [17] is extended and an *associative memory scheme* is proposed for GAs in dynamic environments. For this associative memory scheme, when the best solution of the population is stored into the memory, the current environmental information, the *allele distribution vector*, is also stored in the memory and associated with the best solution. When the environment changes, the stored environmental information associated with the best re-evaluated memory solution is used to create new individuals into the population. Based on the dynamic problem generator proposed in [16, 18], a series of DOPs with different environmental dynamics are constructed as the test bed and experiments are carried out to compare the performance of the proposed associative memory scheme with traditional direct memory scheme for GAs in dynamic environments. Based on the experimental results we analyze the strength and weakness of the associative memory over direct memory for GAs in dynamic environments.

2 Overview of Memory Schemes

The standard GA, denoted *SGA* in this paper, maintains and evolves a population of candidate solutions through selection and variation. New populations are generated by first probabilistically selecting relatively fitter individuals from the current population and then performing crossover and mutation on them to create new off-springs. This process continues until some stop condition becomes true, e.g., the maximum allowable number of generations t_{max} is reached.

Usually, with the iteration of SGA, individuals in the population will eventually converge to the optimal solution(s) in stationary environments due to the pressure of selection. Convergence at a proper pace, instead of pre-mature, may be beneficial and is expected for GAs to locate expected solutions for stationary optimization problems. However, convergence becomes a big problem for GAs in dynamic environments because it deprives the population of genetic diversity. Consequently, when change occurs, it is hard for GAs to escape from the optimal solution of the old environment. Hence, additional approaches, e.g., memory schemes, are required to adapt GAs to the new environment.

The basic principle of memory schemes is to, implicitly or explicitly, store useful information from the current environment and reuse it later in new environments. Implicit memory schemes for GAs in dynamic environments depend on redundant representations to store useful information for GAs to exploit during the run [6, 9, 12]. In contrast, explicit memory schemes make use of precise representations but split an extra storage space where useful information from current generation can be explicitly stored and reused later [2, 10, 15].

For explicit memory there are three technical considerations: what to store in the memory, how to update the memory, and how to retrieve the memory. For the first aspect, usually good solutions are stored in the memory and reused directly when change occurs. This is called *direct memory scheme*. It is also interesting to store environmental information as well as good solutions in the memory and reuse the environmental information when change occurs [8, 13, 17]. This is called *associative memory scheme*, see Section 3 for more information. For the second consideration, since the memory space is limited, it is necessary to update memory solutions to make room for new ones. A general strategy is to select one memory point to be replaced by the best individual from the population. As to which memory point should be updated, there are several strategies [2]. For example, the most similar strategy replaces the memory point that is the closest to the best individual from the population. For the memory retrieval, a natural strategy is to use the best individual(s) in the memory to replace the worst individual(s) in the population. This can be done periodically or only when the environment change is detected.

The GA with the direct memory scheme studied in this paper is called *direct memory GA* (DMGA). DMGA (and other memory based GAs in this study) uses a randomly initialized memory of size $m = 0.1 * n$ (n is the total population size). When the memory is due to update, if any of the randomly initialized points still exists in the memory, the best individual of the population will replace one of them randomly; otherwise, it will replace the closest memory point if it is better (the most similar memory updating strategy). Instead of updating the memory in a fixed time interval, the memory in DMGA is updated in a stochastic time pattern as follows. Suppose the memory is updated at generation t, the next memory updating time t_M is given by: $t_M = t + rand(5, 10)$. This dynamic time pattern can smooth away the potential effect that the environmental change period coincides with the memory updating period (e.g., the memory is updated whenever the environment changes).

The memory in DMGA is re-evaluated every generation to detect environmental changes. The environment is detected as changed if at least one individual in the memory has been detected changed its fitness. If an environment change is detected, the memory is merged with the old population and the best $n - m$ individuals are selected as an interim population to undergo standard genetic operations for a new population while the memory remains unchanged.

3 Associative Memory for Genetic Algorithms

As mentioned before, direct memory schemes only store good solutions in the memory and directly reuse the solutions (e.g., combining them with the current population) when change occurs. In fact, in addition to good solutions we can also store current environmental information in the memory. For example, Ramsey and Greffenstette [13] studied a GA for robot control problem, where good candidate solutions are stored in a permanent memory together with information about the current environment the robot is in. When the robot incurs a new environment that is similar to a stored environment instance, the associated stored controller solution is re-activated. This scheme was reported to significantly improve the robot's performance in dynamic environments. In [17, 19], a memory scheme was proposed into PBIL algorithms for DOPs, where the working probability vector is also stored and associated with the best sample it creates in the memory. When the environment is detected changed, the stored probability vector associated with the best re-evaluated memory sample is extracted to compete with the current working probability vector to become the future working probability vector for creating new samples.

The idea in [17, 19] can be extended to GAs for DOPs. That is, we can store environmental information together with good solutions in the memory for later reuse. Here, the key thing is how to represent current environment. As mentioned before, given a problem in certain environment the individuals in the population of a GA will eventually converge toward the optimum of the environment when the GA progress its searching. The convergence information, i.e., *allele distribution* in the population, can be taken as the natural representation of current environment. Each time when the best individual of the population is stored in the memory, the statistics information on the allele distribution for each locus, the *allele distribution vector*, can also be stored in the memory and associated with the best individual.

The pseudo-code for the GA with the associative memory, called *associative memory GA* (AMGA), is shown in Fig. 1. Within AMGA, a memory of size $m = 0.1 * n$ is used to store solutions and environmental information. Now each memory point consists of a pair $< S, \boldsymbol{D} >$, where S is the stored solution and $\boldsymbol{D}$ is the associated allele distribution vector. For binary encoding (as per this paper), the frequency of ones over the population in a gene locus can be taken as the allele distribution for that locus.

As in DMGA, the memory in AMGA is re-evaluated every generation. If an environmental change is detected, the allele distribution vector of the best memory point $< S_M(t), \boldsymbol{D}_M(t) >$, i.e., the memory point with its solution $S_M(t)$ having the highest re-evaluated fitness, is extracted. And a set of $\alpha * (n - m)$ new individuals are created from this allele distribution vector $\boldsymbol{D}_M(t)$ and randomly swapped into the population. Here, the parameter $\alpha \in [0.0, 1.0]$, called *associative factor*, determines the number of new individuals and hence the impact of the associative memory to the current population. Just as sampling a probability vector in PBIL algorithms [1], a new individual $S = \{s_1, \cdots, s_l\}$ is created by $\boldsymbol{D}_M(t) = \{d_1^M, \cdots, d_l^M\}$ (l is the encoding length) as follows:

$$s_i = \begin{cases} 1, & \text{if } rand(0.0, 1.0) < d_i^M \\ 0, & otherwise \end{cases} \tag{1}$$

The memory replacement strategy in AMGA is similar to that in DMGA. When the memory is due to update, if there are still any randomly initialized memory points in the memory, a random one will be replaced by $<S_P(t), \boldsymbol{D}_P(t)>$, where $S_P(t)$ and $\boldsymbol{D}_P(t)$ are the best individual and allele distribution vector of the current population respectively; otherwise, we first find the memory point $< S_M^c(t), \boldsymbol{D}_M^c >$ with its solution $S_M^c(t)$ closest to $S_P(t)$. If $S_P(t)$ is fitter than $S_M^c(t)$, i.e., $f(S_P(t)) > f(S_M^c(t))$, the memory point is replaced by $<S_P(t), \boldsymbol{D}_P(t)>$.

The aforementioned direct and associative memory can be combined into GAs. The GA with hybrid direct and associative memory schemes, denoted *DAMGA*,

```
$t := 0$ and $t_M := rand(5, 10)$
initialize $P(0)$ randomly and empty memory $M(0)$
evaluate population $P(0)$
repeat
  evaluate memory $M(t)$
  if environmental change detected then
    denote the best memory point $<S_M(t), \boldsymbol{D}_M(t)>$
    $I(t)$ := create $\alpha * (n - m)$ individuals from $\boldsymbol{D}_M(t)$
    $P'(t)$ := swap individuals in $I(t)$ into $P(t)$ randomly
    if direct memory combined then    // for DAMGA
      $P'(t)$ := retrieveBestMembersFrom($P'(t), M(t)$)
  else $P'(t) := P(t)$
  if $t = t_M$ then $t_M := t + rand(5, 10)$  // time to update memory
    denote the best individual in $P'(t)$ by $S_P(t)$
    extract the allele distribution vector $\boldsymbol{D}_P(t)$ from $P'(t)$
    if still any random point in memory then
      replace a random one by $<S_P(t), \boldsymbol{D}_P(t)>$
    else find memory point $<S_M^c(t), \boldsymbol{D}_M^c(t)>$ closest to $<S_P(t), \boldsymbol{D}_P(t)>$
      if $f(S_P(t)) > f(S_M^c(t))$ then $<S_M^c(t), \boldsymbol{D}_M^c> := <S_P(t), \boldsymbol{D}_P(t)>$
  // standard genetic operations
  $P'(t)$ := selectForReproduction($P'(t)$)
  crossover($P'(t), p_c$) // $p_c$ is the crossover probability
  mutate($P'(t), p_m$)  // $p_m$ is the mutation probability
  replace elite from $P(t - 1)$ into $P'(t)$ randomly
  evaluate the interim population $P'(t)$
until terminated = true        // e.g., $t > t_{max}$
```

Fig. 1. Pseudo-code for the AMGA and DAMGA

is also shown in Fig. 1. DAMGA differs from AMGA only as follows. After new individuals have been created and swapped into the population, the original memory solutions $M(t)$ are merged with the population to select $n - m$ best ones as the interim population to go though standard genetic operations.

4 Dynamic Test Environments

The DOP generator proposed in [16, 18] can construct *random* dynamic environments from any binary-encoded stationary function $f(\boldsymbol{x})$ ($\boldsymbol{x} \in \{0, 1\}^l$) by a bitwise exclusive-or (XOR) operator. We suppose the environment changes every τ generations. For each environmental period k, an XORing mask $\boldsymbol{M}(k)$ is incrementally generated as follows:

$$\boldsymbol{M}(k) = \boldsymbol{M}(k-1) \oplus \boldsymbol{T}(k), \tag{2}$$

where "$\oplus$" is the XOR operator (i.e., $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$) and $\boldsymbol{T}(k)$ is an intermediate binary template randomly created with $\rho \times l$ ones (ρ is a parameter) for environmental period k. For the first period $k = 1$, $\boldsymbol{M}(1)$ is set to a zero vector. Then, the population at generation t is evaluated as below:

$$f(\boldsymbol{x}, t) = f(\boldsymbol{x} \oplus \boldsymbol{M}(k)), \tag{3}$$

where $k = \lceil t/\tau \rceil$ is the environmental period index. With this generator, the parameter τ controls the change speed while $\rho \in (0.0, 1.0)$ controls the severity of environmental changes. Bigger ρ means severer environmental change.

The above generator can be extended to construct *cyclic dynamic environments*[1], see [19], as follows. First, we can generate $2K$ XORing masks $\boldsymbol{M}(0), \cdots,$ $\boldsymbol{M}(2K-1)$ as the *base states* in the search space randomly. Then, the environment can cycle among them in a fixed logical ring. Suppose the environment changes every τ generations, then the individuals at generation t is evaluated as:

$$f(\boldsymbol{x}, t) = f(\boldsymbol{x} \oplus \boldsymbol{M}(I_t)) = f(\boldsymbol{x} \oplus \boldsymbol{M}(k\%(2K))), \tag{4}$$

where $k = \lfloor t/\tau \rfloor$ is the index of current environmental period and $I_t = k\%(2K)$ is the index of the base state the environment is in at generation t.

The $2K$ XORing masks can be generated as follows. First, we construct K binary templates $\boldsymbol{T}(0), \cdots, \boldsymbol{T}(K-1)$ that form a random partition of the search space with each template containing $\rho \times l = l/K$ bits of ones[2]. Let $\boldsymbol{M}(0) = \boldsymbol{0}$ denote the initial state, the other XORing masks are generated iteratively as:

$$\boldsymbol{M}(i+1) = \boldsymbol{M}(i) \oplus \boldsymbol{T}(i\%K), i = 0, \cdots, 2K-1 \tag{5}$$

The templates $\boldsymbol{T}(0), \cdots, \boldsymbol{T}(K-1)$ are first used to create K masks till $\boldsymbol{M}(K) =$ $\boldsymbol{1}$ and then orderly reused to construct another K XORing masks till $\boldsymbol{M}(2K) =$ $\boldsymbol{M}(0) = \boldsymbol{0}$. The Hamming distance between two neighbour XORing masks is the same and equals $\rho \times l$. Here, $\rho \in [1/l, 1.0]$ is the distance factor, determining the number of base states.

We can further construct *cyclic dynamic environments with noise* [19] as follows. Each time the environment is about to move to a next base state $\boldsymbol{M}(i)$, noise is applied to $\boldsymbol{M}(i)$ by flipping it bitwise with a small probability p_n.

In this paper, the 100-bit *OneMax* function is selected as the base stationary function to construct dynamic test environments. *OneMax* function aims to maximize the number of ones in a binary string. Three kinds of dynamic environments, cyclic, cyclic with noise, and random, are constructed from the base function using the aforementioned dynamic problem generator. For cyclic environments with noise, the parameter p_n is set to 0.05. For each dynamic environment, the landscape is periodically changed every τ generations during the

[1] For the convenience of description, we differentiate the environmental changing periodicality in time and space by wording *periodical* and *cyclic* respectively. The environment is said to be *periodically* changing if it changes in a fixed time interval, e.g., every certain GA generations, and is said to be *cyclicly* changing if it visits several fixed states in the search space in certain order repeatedly.

[2] In the partition each template $\boldsymbol{T}(i)$ ($i = 0, \cdots, K-1$) has randomly but exclusively selected $\rho \times l$ bits set to 1 while other bits set to 0. For example, $\boldsymbol{T}(0) = 0101$ and $\boldsymbol{T}(1) = 1010$ form a partition of the 4-bit search space.

run of an algorithm. In order to compare the performance of algorithms in different dynamic environments, the parameters τ is set to 10, 25 and 50 and ρ is set to 0.1, 0.2, 0.5, and 1.0 respectively. Totally, a series of 36 DOPs, 3 values of τ combined with 4 values of ρ under three kinds of dynamic environments, are constructed from the stationary *OneMax* function.

5 Experimental Study

5.1 Experimental Design

Experiments were carried out to compare the performance of GAs on the dynamic test environments. All GAs have the following generator and parameter settings: tournament selection with tournament size 2, uniform crossover with $p_c = 0.6$, bit flip mutation with $p_m = 0.01$, elitism of size 1, and population size $n = 100$ (including memory size $m = 10$ if used). In order to test the effect of the associative factor α on the performance of AMGA and DAMGA, α is set to 0.2, 0.6, and 1.0 respectively. And the corresponding GAs are reported as α-AMGA and α-DAMGA in the experimental results respectively.

For each experiment of a GA on a DOP, 50 independent runs were executed with the same set of random seeds. For each run 5000 generations were allowed, which are equivalent to 500, 200 and 100 environmental changes for $\tau = 10$, 25 and 50 respectively. For each run the best-of-generation fitness was recorded every generation. The overall performance of a GA on a problem is defined as:

$$\overline{F}_{BOG} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BOG_{ij}}\right), \tag{6}$$

where $G = 5000$ is the total number of generations for a run, $N = 50$ is the total number of runs, and $F_{BOG_{ij}}$ is the best-of-generation fitness of generation i of run j. The off-line performance $\overline{F}_{BOG}$ is the best-of-generation fitness averaged over 50 runs and then averaged over the data gathering period.

5.2 Experimental Results and Analysis

Experiments were first carried out to compare the performance of SGA, DMGA and α-AMGAs under different dynamic environments. The experimental results regarding SGA, DMGA and α-AMGAs are plotted in Fig. 2. The major statistical results of comparing GAs by one-tailed t-test with 98 degrees of freedom at a 0.05 level of significance are given in Table 1. In Table 1, the t-test result regarding *Alg.* 1 − *Alg.* 2 is shown as "+", "−", "s+" and "s−" when *Alg.* 1 is insignificantly better than, insignificantly worse than, significantly better than, and significantly worse than *Alg.* 2 respectively. From Fig. 2 and Table 1 several results can be observed.

First, both DMGA and AMGAs perform significantly better than SGA on most dynamic problems. This result validates the efficiency of introducing memory schemes into GAs in dynamic environments. Viewing from left to right in

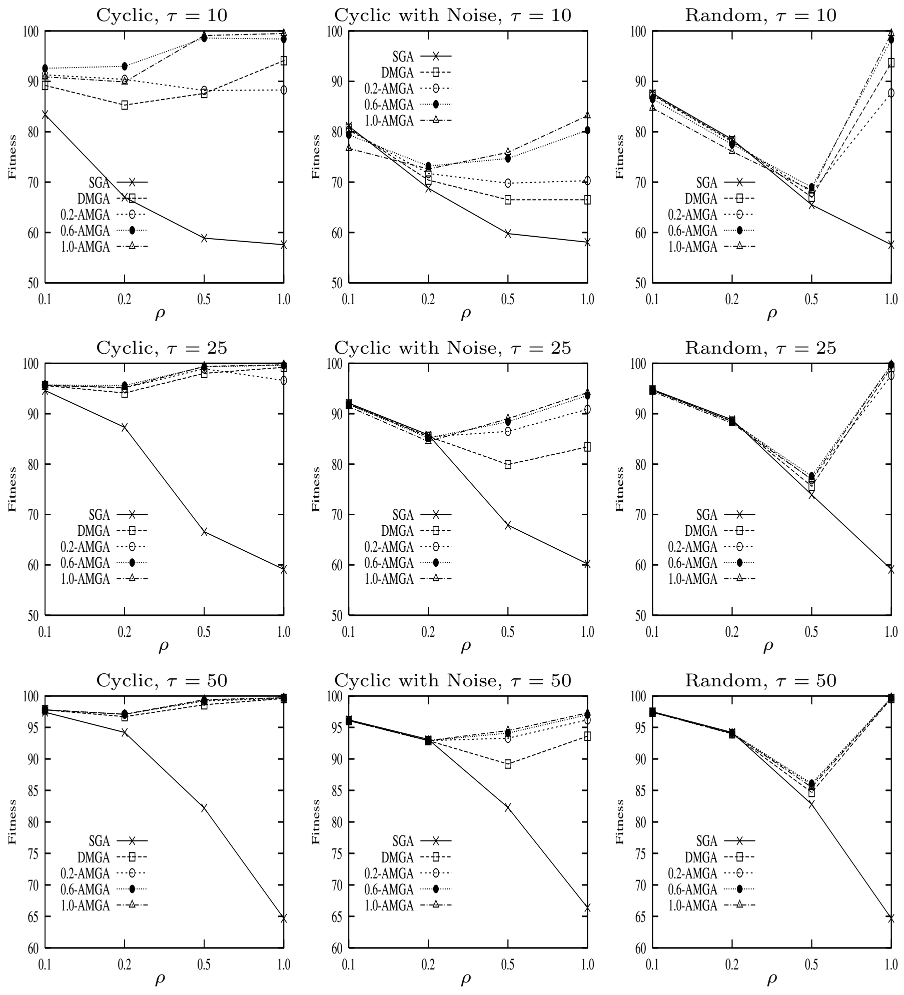


Fig. 2. Experimental results of SGA, DMGA, and α-AMGAs

Fig. 2, it can be seen that both DMGA and AMGAs achieve the largest performance improvement over SGA in cyclic environments. For example, when $\tau = 10$ and $\rho = 0.5$, the performance difference of DMGA over SGA, $\overline{F}_{BOG}(DMGA) - \overline{F}_{BOG}(SGA)$, is $87.6 - 58.9 = 28.7$, $66.5 - 59.8 = 6.7$, and $67.0 - 65.5 = 1.5$ under cyclic, cyclic with noise, and random environments respectively. This result indicates that the effect of memory schemes depends on the cyclicity of dynamic environments. When the environment changes randomly and slightly (i.e., ρ is small), both DMGA and AMGAs are beaten by SGA. This is because under these conditions, the environment is unlikely to return to a previous state that is memorized by the memory scheme. And hence inserting stored solutions or creating new ones according to the stored allele distribution vector may mislead or slow down the progress of the GAs.

Table 1. The t-test results of comparing SAG, DMGA and α-AMGAs

t-test Result	Cyclic				Cyclic with Noise				Random			
$\tau = 10$, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
DMGA − SGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
0.2-AMGA − DMGA	$s+$	$s+$	$+$	$s-$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s-$
0.6-AMGA − DMGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
1.0-AMGA − DMGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
0.6-AMGA − 0.2-AMGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
1.0-AMGA − 0.6-AMGA	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s-$	$s+$
$\tau = 25$, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
DMGA − SGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
0.2-AMGA − DMGA	$s+$	$s+$	$s+$	$s-$	$-$	$s-$	$s+$	$s+$	$-$	$-$	$s+$	$s-$
0.6-AMGA − DMGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
1.0-AMGA − DMGA	$-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
0.6-AMGA − 0.2-AMGA	$-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
1.0-AMGA − 0.6-AMGA	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s-$	$s+$
$\tau = 50$, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
DMGA − SGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$
0.2-AMGA − DMGA	$s+$	$s+$	$s+$	$s+$	$+$	$+$	$s+$	$s+$	$-$	$-$	$s+$	$s+$
0.6-AMGA − DMGA	$s+$	$s+$	$s+$	$s+$	$-$	$+$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$
1.0-AMGA − DMGA	$s+$	$s+$	$s+$	$s+$	$-$	$-$	$s+$	$s+$	$-$	$+$	$s+$	$s+$
0.6-AMGA − 0.2-AMGA	$s+$	$+$	$s+$	$s+$	$-$	$-$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$
1.0-AMGA − 0.6-AMGA	$-$	$-$	$s+$	$+$	$+$	$-$	$s+$	$s+$	$-$	$s-$	$s-$	$s+$

Second, comparing AMGAs over DMGA, it can be seen that AMGAs outperform DMGA on many DOPs, especially under cyclic environments. This happens because the extracted memory allele distribution vector is much stronger than the stored memory solutions in adapting the GA to the new environment. However, when ρ is small and the environment changes randomly, AMGAs are beaten by DMGA for most cases, see the t-test results regarding α-AMGA – DMGA. This is because under these environments the negative effect of the associative memory in AMGAs may weigh over the direct memory in DMGA.

In order to better understand the performance of GAs, the dynamic behaviour of GAs regarding best-of-generation fitness against generations on dynamic *OneMax* functions with $\tau = 10$ and $\rho = 0.5$ under different cyclicity of dynamic environments is plotted in Fig. 3. In Fig. 3, the first and last 10 environmental changes (i.e., 100 generations) are shown and the data were averaged over 50 runs. From Fig. 3, it can be seen that, under cyclic and cyclic with noise environments, after several early stage environmental changes, the memory schemes start to take effect to maintain the performance of DMGA and AMGAs at a much higher fitness level than SGA. And the associative memory in AMGAs works better than the direct memory in DMGA, which can be seen in the late stage behaviour of GAs. Under random environments the effect of memory schemes is greatly deduced where all GAs behave almost the same and there is no clear view of the memory schemes in DMGA and AMGAs.

Third, when examining the effect of α on AMGA's performance, it can be seen that 0.6-AMGA outperforms 0.2-AMGA on most dynamic problems, see the t-test results regarding 0.6-AMGA – 0.2-AMGA. This is because increasing the value of α enhances the effect of associative memory for AMGA. However, 1.0-AMGA is beaten by 0.6-AMGA on many cases, especially when ρ is small, see the t-test results regarding 1.0-AMGA – 0.6-AMGA. When $\alpha = 1.0$, all the

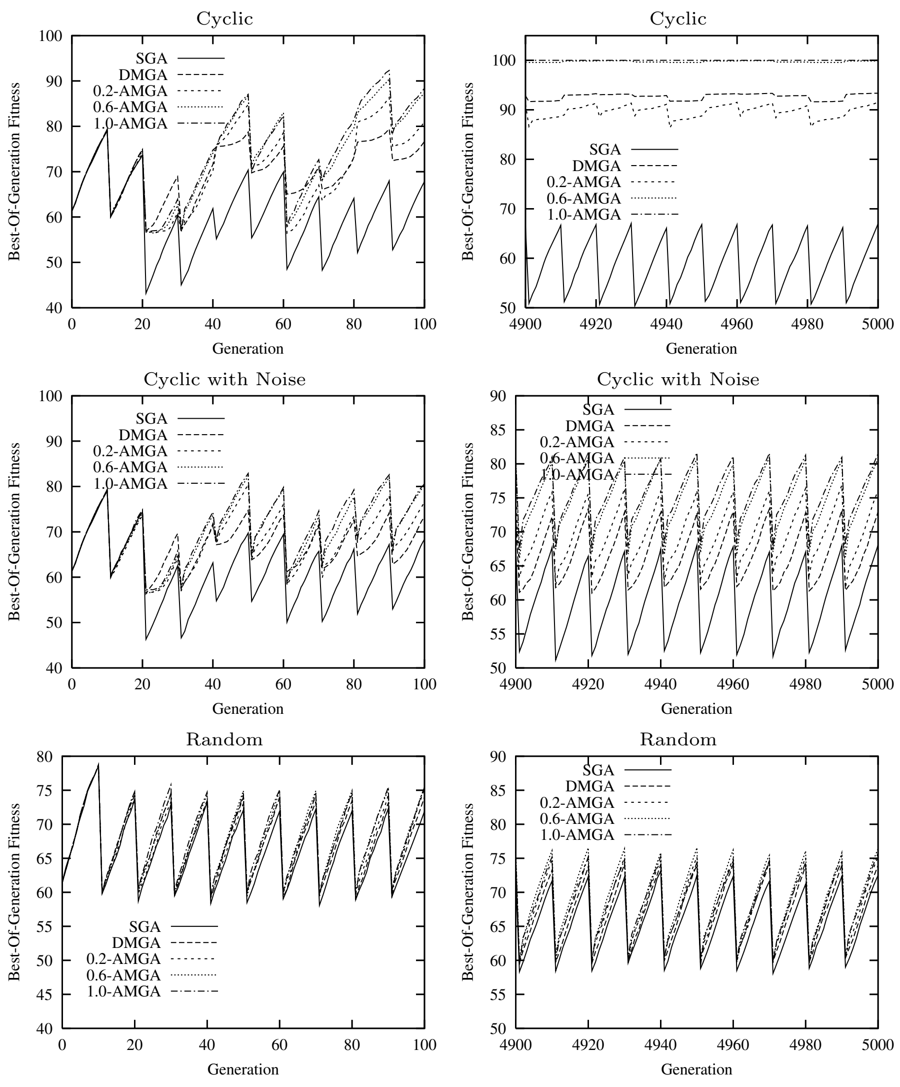


Fig. 3. Dynamic behaviour of GAs during the (*Left Column*) early and (*Right Column*) late stages on dynamic *OneMax* functions with $\tau = 10$ and $\rho = 0.5$

individuals in the population are replaced by the new individuals created by the re-activated memory allele distribution vector when change occurs. This may be disadvantageous. Especially, when ρ is small, the environment changes slightly and good solutions of previous environment are likely also good for the new one. It is better to keep some of them instead of discarding them all.

In order to test the effect of combining direct memory with associative memory into GAs for DOPs, experiments were further carried out to compare the performance of DAMGAs over AMGAs. The relevant t-test results are presented in Table 2, from which it can be seen that DAMGAs outperform AMGAs under most dynamic environments. However, the experiments (not shown here) indicate the performance improvement of α-DAMGA over α-AMGA is relatively small in comparison with the performance improvement of α-AMGA over SGA.

Table 2. The t-test results of comparing α-AMGAs and α-DAMGAs

t-test Result	Cyclic				Cyclic with Noise				Random			
$\tau = 10$, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
0.2-DAMGA − 0.2-AMGA	$s+$	$s+$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$	$-$	$+$	$s+$	$s+$
0.6-DAMGA − 0.6-AMGA	$s+$	$+$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$	$+$	$+$	$s+$	$s+$
1.0-DAMGA − 1.0-AMGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$
$\tau = 25$, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
0.2-DAMGA − 0.2-AMGA	$s+$	$s+$	$s+$	$s+$	$-$	$+$	$s+$	$s+$	$+$	$-$	$s+$	$s+$
0.6-DAMGA − 0.6-AMGA	$s+$	$+$	$s+$	$s+$	$+$	$-$	$+$	$s+$	$+$	$-$	$s+$	$s+$
1.0-DAMGA − 1.0-AMGA	$+$	$s+$	$s+$	$s+$	$+$	$s+$	$+$	$+$	$s+$	$+$	$s+$	$s+$
$\tau = 50$, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
0.2-DAMGA − 0.2-AMGA	$+$	$+$	$-$	$s+$	$+$	$-$	$s+$	$s+$	$+$	$s+$	$s+$	$s+$
0.6-DAMGA − 0.6-AMGA	$+$	$+$	$s+$	$s+$	$-$	$+$	$+$	$s+$	$-$	$-$	$s+$	$s+$
1.0-DAMGA − 1.0-AMGA	$+$	$+$	$+$	$s+$	$-$	$-$	$-$	$+$	$+$	$-$	$s+$	$s+$

6 Conclusions and Discussions

This paper investigates the introduction of an associative memory scheme into GAs for dynamic optimization problems. Within this memory scheme, the allele distribution information is taken as the representation of the current environment that GAs have searched. The allele distribution vector is stored together with the best member of the current population in the memory. When the environmental change is detected, the stored allele distribution vector that is associated with the best re-evaluated memory solution is extracted to create new individuals into the population. A series of dynamic problems were systematically constructed, featuring three kinds of dynamic environments: cyclic, cyclic with noise, and random. Based on this test platform experimental study was carried out to test the proposed associative memory scheme.

From the experimental results, the following conclusions can be drawn on the dynamic test environments. First, memory schemes are efficient to improve the performance of GAs in dynamic environments and the cyclicity of dynamic environments greatly affect the performance of memory schemes for GAs in dynamic environments. Second, generally speaking the proposed associative memory scheme outperforms traditional direct memory scheme for GAs in dynamic environments. Third, the associative factor has an important impact on the performance of AMGAs. Setting α to a medium value, e.g., 0.6, seems a good choice for AMGAs. Fourth, combining the direct scheme with the associative memory scheme may further improve GA's performance in dynamic environments.

For future work, comparing the memory scheme investigated with implicit memory schemes is now under investigation. And it is also interesting to further investigate the interactions between the associative memory scheme and other approaches, such as multi-population and diversity approaches, for GAs in dynamic environments.

References

1. S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. *Tech. Report CMU-CS-94-163*, Carnegie Mellon University, 1994.
2. J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. *Proc. of the 1999 Congr. on Evol. Comput.*, vol. 3, pp. 1875-1882, 1999.
3. J. Branke, T. Kaußler, C. Schmidth, and H. Schmeck. A multi-population approach to dynamic optimization problems. *Proc. of the Adaptive Computing in Design and Manufacturing*, pp. 299-308, 2000.
4. J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, 2002.
5. H. G. Cobb and J. J. Grefenstette. Genetic algorithms for tracking changing environments. *Proc. of the 5th Int. Conf. on Genetic Algorithms*, pp. 523-530, 1993.
6. D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, pp. 59-68, 1987.
7. J. J. Grefenstette. Genetic algorithms for changing environments. *Proc. of the 2nd Int. Conf. on Parallel Problem Solving from Nature*, pp. 137-144, 1992.
8. A. Karaman, S. Uyar, and G. Eryigit. The memory indexing evolutionary algorithm for dynamic environments. *EvoWorkshops 2005*, LNCS 3449, pp. 563-573, 2005.
9. E. H. J. Lewis and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. *Proc. of the 5th Int. Conf. on Parallel Problem Solving from Nature*, pp. 139-148, 1998.
10. N. Mori, H. Kita and Y. Nishikawa. Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. *Proc. of the 7th Int. Conf. on Genetic Algorithms*, pp. 299-306, 1997.
11. R. W. Morrison and K. A. De Jong. Triggered hypermutation revisited. *Proc. of the 2000 Congress on Evol. Comput.*, pp. 1025-1032, 2000.
12. K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimisation. *Proc. of the 6th Int. Conf. on Genetic Algorithms*, 1997.
13. C. L. Ramsey and J. J. Greffenstette. Case-based initializtion of genetic algorithms. *Proc. of the 5th Int. Conf. on Genetic Algorithms*, 1993.
14. A. Simões and E. Costa. An immune system-based genetic algorithm to deal with dynamic environments: diversity and memory. *Proc. of the 6th Int. Conf. on Neural Networks and Genetic Algorithms*, pp. 168-174, 2003.
15. K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. *Proc. of the 1999 Congress on Evol. Comput.*, pp. 1843-1850, 1999.
16. S. Yang. Non-stationary problem optimization using the primal-dual genetic algorithm. *Proc. of the 2003 IEEE Congress on Evol. Comput.*, vol. 3, pp. 2246-2253, 2003.
17. S. Yang. Population-based incremental learning with memory scheme for changing environments. *Proc. of the 2005 Genetic and Evol. Comput. Conference*, vol. 1, pp. 711-718, 2005.
18. S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, vol. 9, no. 11, pp. 815-834, 2005.
19. S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. Submitted to *IEEE Trans. on Evol. Comput.*, 2005.

Bayesian Optimization Algorithms for Dynamic Problems

Miloš Kobliha[1], Josef Schwarz[1], and Jiří Očenášek[2]

[1] Brno University of Technology, Faculty of Information Technology,
Department of Computer Systems, Božetěchova 2, Brno 612 66, CZ
Tel.: +420-5-41141210, Fax: +420-5-41141270
{kobliha, schwarz}@fit.vutbr.cz
[2] Kimotion Technologies, Leuvensesteenweg 200,
Boutersem B-3370, Belgium

Abstract. This paper is an experimental study investigating the capability of Bayesian optimization algorithms to solve dynamic problems. We tested the performance of two variants of Bayesian optimization algorithms – Mixed continuous-discrete Bayesian Optimization Algorithm (MBOA), Adaptive Mixed Bayesian Optimization Algorithm (AMBOA) – and new proposed modifications with embedded Sentinels concept and Hypervariance. We have compared the performance of these variants on a simple dynamic problem – a time-varying function with predefined parameters. The experimental results confirmed the benefit of Sentinels concept and Hypervariance embedded into MBOA algorithm for tracking a moving optimum.

1 Introduction

Evolutionary Algorithms (EAs) have been widely utilized to solve stationary optimization problems. But many real optimization problems are actually dynamic. In case of dynamic problems the fitness function, parameters and environmental conditions may change over time. Most methods for handling dynamic problems encourage higher diversity of population than conventional EAs does. The survey of main techniques in the field is described in [1, 2]. The efficient approach is based on Triggered Hypermutation, which uses two different scales of mutation probability – one for stationary state (0.001) and another for nonstationary state (0.3). For achieving sufficient population diversity, Morrison [2] proposed the concept of Sentinels – subpopulation of individuals in the population not modified by the reproduction process. These "not moving" individuals are also able to provide the detection of environmental changes. A very competent overview of the current methods for dynamic problem optimization is published in [3].

Our goal is to test the capability of the new versions of BOA algorithms including the phenomenon of Hypervariance and Sentinels. The rest of the paper is organized as follows. Section 2 explains MBOA and AMBOA algorithms and the newly proposed modifications in more detail, Section 3 presents experimental results; conclusions and future works are presented in Section 4.

2 BOA Algorithms and Dynamic Optimization

2.1 MBOA Algorithm

MBOA explores the search space by sampling a probability distribution that is developed during the optimization. It works with a population of N candidate solutions/individuals. In each generation, typically the $N/2$ fittest individuals ("parents") are used for the model building and $N/2$ new solutions ("offspring") are generated from the model. These offspring individuals are evaluated and incorporated into the original population, replacing some of the old ones. This process is repeated until the termination criteria are met. More implementation details can be found in [4].

Let us denote the i-th variable as X_i and the set of variables that influence X_i as Π_i. The overall probabilistic model $f(X_0, \ldots, X_{n-1})$ is defined as a product of conditional distributions $p(X_i|\Pi_i)$. MBOA constructs each $p(X_i|\Pi_i)$ by measuring correlations between variable X_i and variables Π_i and by decomposing the domain of $p(X_i|\Pi_i)$ into axis-parallel partitions where X_i is assumed to be decorrelated from Π_i. In each partition the distribution of X_i can be approximated by the following Gaussian kernel:

$$f(X_i \mid \Pi_i) = \frac{1}{\left|\{x_i\}_j\right|} \sum_{\forall \mu \in \{x_i\}_j} N(\mu, \eta_g^2 \sigma_{ij}^2), \tag{1}$$

where $\{x_i\}_j \subset \Re$ denotes the set of realizations of variable X_i among the individuals from parent population that traverse to j-th partition, and $|\{x_i\}_j|$ denotes the number of such individuals. The scaling factor η_g is equal to 1 in MBOA. All kernels in the same leaf have the same height $1/|\{x_i\}_j|$ and the same width σ_{ij}:

$$\sigma_{ij} = \frac{\max\{x_i\}_j - \min\{x_i\}_j}{|\{x_i\}_j| - 1}. \tag{2}$$

2.2 AMBOA with Variance Adaptation

In order to prevent optimizer from premature convergence, the MBOA with variance adaptation (AMBOA) was introduced in [5]. An overall scaling factor η_g controls the width of Gaussian kernels (see Eq.1). For $N/2$ offspring individuals (with N_{succ} successes and N_{fail} failures), the total change of factor in the $(g+1)$-th generation is:

$$\eta_{g+1} = \eta_g \alpha^{N_{succ}} \alpha^{N_{fail}\frac{p}{p-1}}, \tag{3}$$

where p denotes the desired success rate (for $N_{succ}/(N_{succ} + N_{fail})=p$ it holds $\eta_{g+1}=\eta_g$). The choice of α determines how fast the desired success rate is achieved. For increasing α the adaptation is faster, but also more sensitive to oscillations. In our experiments, we choose $\alpha = e^{4/N}$ and $p=0.05+0.3/n^{-1/2}$, in accordance with [5].

2.3 AMBOA and MBOA Algorithms with Hypervariance

Firstly, we proposed a new version – Adaptive Hypervariance Mixed Bayesian Optimization Algorithm (AHMBOA) with a new interpretation of η, (see Eq.1).The value of η is set to a suitably large value in generation immediately after the change of fitness function. In next generations the value of η is set back to the value before the change. Secondly, we designed Hypervariance Mixed Bayesian Optimization Algorithm (HMBOA) which is derived from AHMBOA by skipping the adaptation of η during the optimization process.

2.4 The Embedded Sentinels

Sentinels are permanent members of the population, uniformly distributed through the search space. They participate on the generation of offspring, but they are not replaced by new individuals during the tournament replacement process. In order to distribute Sentinels uniformly in the search space we used the well known formula published in [2].

3 Experimental Results

Our goal is to test and compare the ability of the proposed BOA versions to track the optimum in dynamic environments. We used a simple dynamic test problem which can be interpreted as a problem with changing fitness peak location. The performance is measured by the Collective Mean Fitness published in [2]:

$$E_C = \sum_{g=1}^{G}\left(\sum_{m=1}^{M}(F_{BG})/M\right)/G\,, \tag{4}$$

where E_C is the Collective Mean Fitness, F_{BG} is the fitness value of the best solution in current generation, M is the number of runs, and G is the number of generations. We used four levels of Sentinels in population (0, 30, 50, and 70 percent). The test function *F(X,g)* is defined as a normalized sum of moving peaks:

$$F(X,g) = F(x_1,\ldots,x_8,g) = \frac{1}{8}\sum_{i=1}^{8} f(x_i,g)\ , \tag{5}$$

with each moving peak $f(x_i,g)$ having the sinusoidal form

$$f(x_i,g) = \begin{cases} \sin(x_i - \lfloor g/T \rfloor \cdot k \cdot \pi) & 0 < (x_i - \lfloor g/T \rfloor \cdot k \cdot \pi) < \pi \\ 0 & otherwise \end{cases}\ , \tag{6}$$

where $x_i \in <0,80\pi>$, g is the generation number, T is the movement period and $k \in \{0.5,4\}$ is the step-scale. Thus, the shift of the sinus function is done using the staircase floor function which represents the mobility of the sinusoidal peak.

We have used three control parameters of experiments: the percentage S of Sentinels in the population, two movement parameters which set the movement mode – the movement period T and the step-scale k. The Collective Mean Fitness E_C (see Eq. 4) is calculated for M=10 runs, permanent number of generations (G=1000) and F_{BG} equal to best *F(X,g)*. The movement period was in the range $T \in <40,200>$.

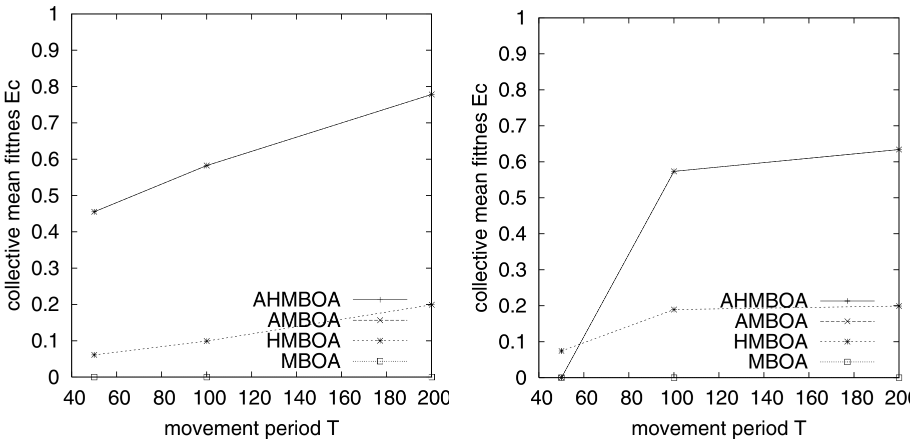


Fig. 1. The effect of movement period T for S=0%, the step-scale k=0.5 (left), and k=4.0 (right)

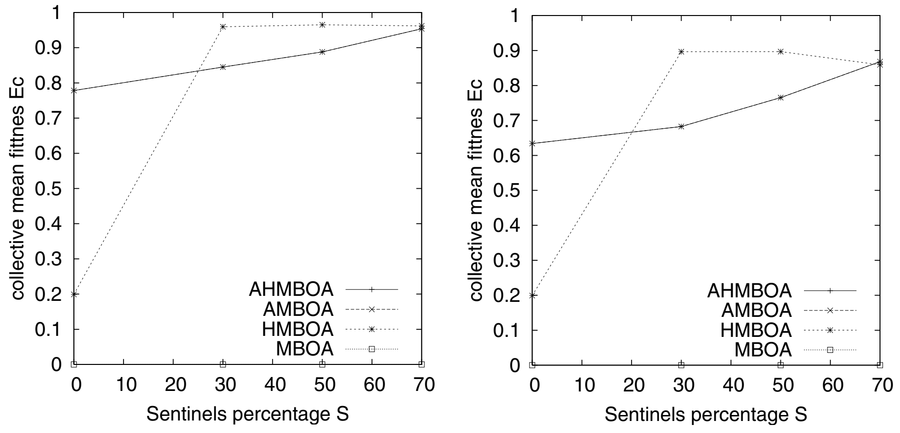


Fig. 2. The effect of Sentinels for T=200, the step-scale k=0.5 (left), and k=4.0 (right)

In Fig. 1 the dependency of E_C on the movement period T for two values of step-scale k and zero percentage of Sentinels is presented. It is clear that E_C is increasing for longer movement periods. AMBOA and AHMBOA outperform HMBOA. The effect of variance adaptation is significantly better than the effect of Hypervariance applied only during one critical generation after the change of the fitness function. Let us note that MBOA was not able to track the optimum in any experiment.

In Fig. 2 the positive influence of embedded Sentinels is demonstrated for all tested algorithms in case of movement period T equal to 200 generations. It is evident that the HMBOA algorithm outperformed AHMBOA and AMBOA for the Sentinel percentage value greater than 30. This can be explained by a phenomenon when the influence of Sentinels partly resulted in limited exploitation of search is balanced by the presence of Hypervariance.

4 Conclusions

In the paper the performance of two variants of Bayesian optimization algorithms was tested – MBOA algorithm developed for optimization of mixed continuous discrete problems and AMBOA algorithm which extends MBOA with variance adaptation. Both algorithms confirmed its applicability to dynamic environment, but with the limited minimal period of the change of fitness function.

That is why we proposed a new extension of MBOA and AMBOA algorithm with embedded Sentinel concept. This technique contributes to the implicit adaptation of the probabilistic model after the change of fitness function. The Sentinels concept together with Hypervariance included in HMBOA algorithm was the best choice resulting in significantly improved performance. The future work will be focused on testing the performance on more complex benchmarks. We will consider more advanced adaptation scheme for the probabilistic model building using additional information derived from embedded Sentinels.

Acknowledgements. This research has been carried out under the financial support of the research project FR0136/2006/G1 "Evolutionary algorithms for dynamic problems", (Ministry of Education), and the research grant GA102/06/0599 "Methods of polymorphic digital circuit design" (Grant Agency of Czech Republic).

References

1. Branke, J.: Evolutionary Optimization in Dynamic Environment. University of Karlsruhe, Germany, Kluwer Academic Publishers, 2002.
2. Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Springer – Verlag, Germany, 2004, pp.147.
3. Yaochu, J., Branke, J.: Evolutionary optimization in uncertain environments–a survey. IEEE Transactions on Evolutionary Computation, Volume 9, Issue 3, June 2005, pp. 303 – 317.
4. Ocenasek, J., Schwarz, J.: Estimation of Distribution Algorithm for mixed continuous-discrete optimization problems. In: 2nd Euro-International Symposium on Computational Intelligence, IOS Press, Kosice, Slovakia, 2002, pp. 227–232.
5. Ocenasek, J., Kern, S., Hansen, N., Müller, S., Koumoutsakos, P.: A Mixed Bayesian Optimization Algorithm with Variance Adaptation. Parallel Problem Solving From Nature – PPSN VIII, Springer – Verlag, Berlin, 2004, pp. 352–361.

Prudent-Daring vs Tolerant Survivor Selection Schemes in Control Design of Electric Drives

Ferrante Neri[1], Giuseppe L. Cascella[1], Nadia Salvatore[1], Anna V. Kononova[2], and Giuseppe Acciani[1]

[1] Dipatimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via E. Orabona 4, Bari, 70125, Italy
neri@deemail.poliba.it, leocascella@ieee.org, n.salvatore@deemail.poliba.it, acciani@deemail.poliba.it

[2] Laboratory of Theoretical Probabilistic Methods, Yaroslavl' State University, 150000, Sovetskaya st. 14, Yaroslavl', Russia
annk2000@mail.ru

Abstract. This paper proposes and compares two approaches to defeat the noise due the measurement errors in control system design of electric drives. The former is based on a penalized fitness and two cooperative-competitive survivor selection schemes, the latter is based on a survivor selection scheme which makes use of the tolerance interval related to the noise distribution. These approaches use adaptive rules in parameter setting to execute both the explicit and the implicit averaging in order to obtain the noise defeating in the optimization process with a relatively low number of fitness evaluations. The results show that the two approaches differently bias the population diversity and that the first can outperform the second but requires a more accurate parameter setting.

1 Introduction and Problem Description

When an evolutionary algorithm is implemented, the individuals are explicitly or implicitly sorted according to their fitness values in order to perform a parent selection, a survivor selection or to assign a lifetime score. If the evolutionary optimization is performed in a noisy environment, the solutions can be wrongly sorted due to the fitness overestimations and underestimations (see [1] and [2]). This paper proposes and compares two different approaches to treat the noisy environment and shows an application to the control of a Permanent Magnet Synchronous Motor (PMSM) in presence of measurement errors. In Fig. 1 the block diagram of a vector-controlled PMSM drive is shown. For details concerning this control scheme see [3], [4] and [5]. The problem of the control design (self-commissioning) can be formulated as the determination of ten parameters (see Fig. 2) which solve a multi-objective optimization problem in $H \subset \mathbb{R}^{10}$. The performance given by each solution is numerically evaluated through a cost objective function f built up by means of the weighted-sum of $f_{1,j}$, $f_{2,j}$, $f_{3,j}$ and $f_{4,j}$ which respectively measure the speed error at the settling, speed overshoot,

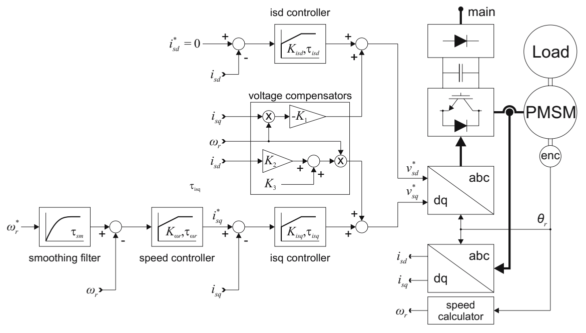


Fig. 1. Motor control system

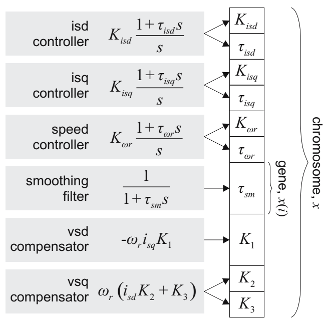


Fig. 2. Candidate solution

speed rise-time, and undesired d-axis-current oscillations for each speed step j of a training test (see [5] for details). Since each single objective function requires a set of measurements the measurement errors affect the objective function f which is therefore noisy.

2 Prudent-Daring and Tolerant Selection Schemes

The Adaptive Prudent-Daring Evolutionary Algorithm (APDEA) executes the minimization of f operating dynamically on both the population size and the number of fitness evaluations (sample size). For each individual, the fitness f is replaced with an "image" fitness function given by $\hat{f} = f_{est} + \frac{b}{n_s^i}$ where the estimated fitness f_{est} [1] is the current average fitness calculated over n_s^i samples (related to the i^{th} individual of the population) and b is a weight coefficient. $\frac{b}{n_s^i}$ is a penalty term which has a big influence for unreliable solutions (n_s^i low) and which progressively tends to have a negligible influence for reliable solutions ($n_s^i \gg 1$). Besides, a maximum number of samples n_s^{max} is established taking into account the features of the noise under examination. An initial sampling of points (see Fig.2) is done at pseudo-random. At the first generation the fitness $\hat{f}$ is calculated (with $n_s^i = 1$) for all the individuals and the coefficient $\xi = \min\left\{1, \left|\frac{\hat{f}_{best} - \hat{f}_{avg}}{\hat{f}_{best}}\right|\right\}$ is determined. $\hat{f}_{best}$ and $\hat{f}_{avg}$ are respectively the best and average fitness values among the individuals of the population. The coefficient ξ is a fitness-based index of the population diversity; it can be seen as a measurement of the state of the convergence of the algorithm (see [5]). In fact if $\xi \approx 1$ there is a high population diversity and therefore the convergence conditions are far; if $\xi \approx 0$ there is a low population diversity and therefore the convergence is approaching. At each subsequent generation μ individuals undergoing crossover are selected according to the ranking selection [6] and the blend crossover [7] is then applied. The mutation probability is then calculated by $p_m = 0.4\,(1 - \xi)$ and the mutation is executed (see for details [5]). The fitness values of the λ offspring individuals are calculated (with $n_s^i = 1$) and the population made up of both parents and offspring ($\mu + \lambda$) undergoes the following survivor selection based on two cooperative-competitive [8] schemes. a) *Prudent* Survivor Selection: the value $S_{pru} = S_{pru}^{f} + S_{pru}^{v}(1-\xi)$ is calculated and the best performing S_{pru} individuals according to $\hat{f}$ are thus selected. S_{pru}^{f} is the minimum size of the prudent population and S_{pru}^{v} is the maximum size of the variable population; b) *Daring* Survivor Selection: the value $S_{dar} = round\left(S_{dar}^{max}\xi\right)$ is calculated and the best performing S_{dar} individuals according to f_{est} are thus selected. S_{dar}^{max} is maximum size of the daring population; c) The prudent and daring populations are merged ($S_{pop} = S_{pru} + S_{dar}$). Thus, the algorithm prudently selects a part of the population taking into account the reliability of the solutions and dares give the surviving chance to those solutions which are not reliable (n_s^i low) but which are promising (f_{est} high). Moreover, the algorithmic logic is based on the idea that for $\xi \approx 1$ (S_{pru} small and S_{dar} big) the fitness values are very different among each other and a wrongly estimated individual coming from the daring survivor selection does not strongly affect the operation of sorting according to the fitness. On the contrary, for $\xi \approx 0$ (S_{pru} big and S_{dar} small) a wrongly estimated individual could significantly affect the operation of sorting according to the fitness. In the last case, to dare introduce an unreliable individual could mean to introduce a wrong direction search [1]. The newly merged population then undergoes a reevaluating cycle: for each individual the value of additional samples $n_s^{add} = round\left(n_s^{max}\frac{(1-\xi)}{n_s^i}\right)$ is calculated and n_s^{add} fitness reevaluations are executed. The values of n_s^i, f_{est} and $\hat{f}$ are then updated. Consequently, the number of reevaluations to be executed on one individual depends on the state of the whole population (i.e. ξ) and on the previous history of the individual (i.e. n_s^i). Finally, the coefficient ξ is updated at the end of each generation.

The Adaptive Tolerant Evolutionary Algorithm (ATEA) assumes that the noise is Gaussian and that its standard deviation has the same constant value in all the points of the domain H under study. Taking into account these hypotheses, the wideness w_{TI} of the Tolerance Interval related to noise has been determined as shown in [9]. The ATEA works on the fitness $\tilde{f} = f_{est}$ [1]. An initial sampling is performed at pseudo-random. The fitness values f of these individuals are determined and the coefficient $\xi = \min\left\{1, \left|\frac{f_{best}-f_{avg}}{f_{best}}\right|\right\}$ is thus calculated. In the generic k^{th} generation the following steps are executed. Selection (μ), blend corossover and mutation occur as for the APDEA. The fitness values related to the offspring newly generated (λ) are thus calculated. The $(\mu+\lambda)$ individuals undergo the Tolerant Survivor Selection consisting of the following. a) The individuals are sorted according to the fitness $\tilde{f}$; b) The population size $S_{pop} = S_{pop}^{f} + S_{pop}^{v}(1-\xi)$ is calculated; c) The individual having position S_{pop} with fitness $\tilde{f}_{S_{pop}}$ is detected and an auxiliary population made up of individuals whose fitness value falls within $\left[\tilde{f}_{S_{pop}} - \frac{w_{TI}}{2}, \tilde{f}_{S_{pop}} + \frac{w_{TI}}{2}\right]$ is created; d) For each individual of this auxiliary population the value $n_s^{add} = round\left(n_s^{max}\frac{(1-\xi)}{n_s^i}\right)$ is calculated and n_s^{add} fitness reevaluations are executed. The values of n_s^i and $\tilde{f}$ are then updated; e) The main population (made up of $(\mu+\lambda)$ individuals) is updated and resorted according to $\tilde{f}$; f) The best performing S_{pop} individuals are saved for the subsequent generation. Finally, the value of ξ is updated according to the formula $\xi = \min\left\{1, \left|\frac{\tilde{f}_{best}-\tilde{f}_{avg}}{\tilde{f}_{best}}\right|\right\}$. The main idea behind the ATEA is that if it is possible to prove that an individual, even if underestimated, is in the top part of the list, it should not be reevaluated; analogously if an individual is so bad that, even if overestimated, is in the bottom part of the list. In other words, when S_{pop} is calculated, the algorithm implicitly divides the population in three categories: individuals surely good, individuals surely bad and individuals which require a more attentive analysis. The individuals surely good or bad do not require additional evaluations; the others require a reevaluation cycle.

3 Numerical Results and Conclusion

Following the procedure described in [9] for Gaussian distribution, the 90% of the fitness samples falls in a tolerance interval with wideness $w_{TI} = 0.1702$ with a probability $\gamma = 0.99$. Both the APDEA and the ATEA have been executed in order to minimize the fitness f with $n_s^{max} = 10$. Concerning the APDEA $S_{pru} \in [40, 160]$, $S_{dar} \in [0, 20]$ and $b = 0.2$; concerning the ATEA $S_{pop} \in [40, 160]$. Also a standard Reevaluating Genetic Algorithm (RGA) employing the same crossover and mutation techniques used for the APDEA and the ATEA has been implemented. This RGA executes the averaging over time [2] with a sample size $n_s^{max} = 10$ for every evaluation and makes use of a standard survivor selection which saves at each generation the prefixed $S_{pop} = 100$ best performing individuals. Each of the three algorithms has been executed 65 times. Each execution has been stopped after 20000 fitness evaluations. Table 1 compares the best performing solutions found by the three methods and shows the best fitness values f, the average best fitness $\bar{f}$ (over the 65 experiments) and the corresponding standard deviation σ. The algorithmic performances and the behavior of ξ for the APDEA and the ATEA are shown in Fig. 3 and in Fig. 4 respectively. The numerical results show that both the APDEA and the ATEA converge faster than the RGA to solutions very similar among each other. Concerning the convergence velocity, the APDEA proved to be more performing than the ATEA. Moreover the APDEA is more general than the ATEA since the latter makes use of the assumption that the noise is Gaussian and with a constant σ in all the domain. On the other hand, the APDEA, unlike the ATEA, requires the setting of b and S_{dar}^{max}. A wrong choice of b would lead to a too strong or too weak penalization in the fitness function. Analogously, S_{dar}^{max} determines the audacity of the algorithm and its wrong setting could lead to either a wrong

Table 1. Solutions and related Fitness

	K_{isd}	τ_{isd}	K_{isq}	τ_{isq}	$K_{\omega r}$	$\tau_{\omega r}$	τ_{sm}	K_1	K_2	K_3	f	$\bar{f}$	σ
RGA	10.99	0.0023	6.66	0.0012	0.243	0.0141	0.0106	0.0019	0.0009	0.1891	0.858	0.867	0.0134
APDEA	11.49	0.0022	6.20	0.0011	0.264	0.0145	0.0109	0.0021	0.0006	0.1901	0.851	0.861	0.0158
ATEA	11.14	0.0021	6.61	0.0013	0.259	0.0132	0.0101	0.0020	0.0008	0.1957	0.854	0.860	0.0120

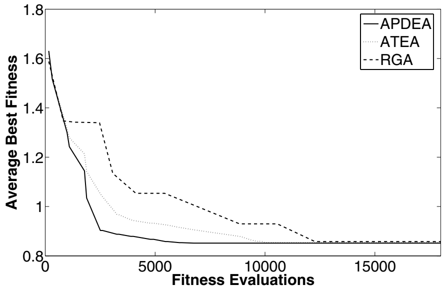


Fig. 3. Performances of the three algorithms

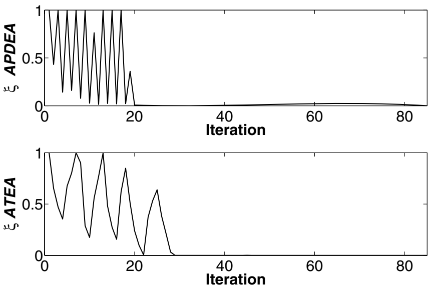


Fig. 4. Behavior of ξ for the APDEA and the ATEA

search direction or an excessive exploitation. Regarding n_s^{max}, that is a parameter common for both APDEA and ATEA, the setting is a much less critical issue. In fact, it can be set as the minimum sample size which describes a proportion of distribution with a given probability (see [9]). Fig. 4 shows that in the case of the APDEA, ξ has high-frequency oscillations before settling down to the value 0. For the ATEA, ξ has less oscillations with low-frequency. Our interpretation of this phenomenon is the following. The APDEA introduces during the daring selection some unreliable solutions before reevaluating them. This behavior leads to an abrupt increasing of the population diversity that is corrected during the survivor selection of the subsequent generation. On the contrary, the ATEA aims to properly sort the candidate solutions and to include for the subsequent generation only the solutions that are surely in the top part of the list. Consequently, the classical recombination and mutation are the ones in charge of the exploration. This logic leads to a milder variation of the population diversity.

References

1. Branke, J.: Evolutionary Optimization in Dynamic Enviroments. Kluwer (2001)
2. Jin, Y., Branke, J.: Evultionary optimization in uncertain enviroments-a survey. IEEE Trans. on Evolutionary Computation **9** (2005) 303–317
3. Krishnan, R.: Electronic Motor Drives: Modeling, Analysis and Control. Prentice Hall, Upper Saddle River, New Jersey, USA (2001)
4. Leonhard, W.: Control of Electrical Drives. 2nd edn. Springer-Verlag (1996)
5. Caponio, A., Cascella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives. To app. IEEE Trans. on System Man and Cybernetics-B, spec. issue Memetic Alg.s (2006)
6. Whitley, D.: The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In: Proc. 3rd Int. Conf. on GAs. (1989) 116–121
7. Eshelman, L.J., Shaffer, J.D.: Real-coded genetic algorithms and interval-schemata. In: Foundations of Genetic Algorithms 2, Morgan Kaufmann (1993) 187–202
8. Ong, Y.S., Keane, A.J.: Meta-lamarkian learning in memetic algorithms. IEEE Trans. on Evolutionary Computation **8** (2004) 99–110
9. NIST: e-handbook of statistical methods. (www.itl.nist.gov/div898/handbook/)

Author Index