Mikhail Prokopenko (Ed.)

Advances in Applied Self-Organizing Systems

Springer

Advanced Information and Knowledge Processing

Also in this series

Gregoris Mentzas, Dimitris Apostolou, Andreas Abecker and Ron Young
Knowledge Asset Management 1-85233-583-1

Michalis Vazirgiannis, Maria Halkidi and Dimitrios Gunopulos
Uncertainty Handling and Quality Assessment in Data Mining 1-85233-655-2

Asunción Gómez-Pérez, Mariano Fernández-López and Oscar Corcho
Ontological Engineering 1-85233-551-3

Arno Scharl (Ed.)
Environmental Online Communication 1-85233-783-4

Shichao Zhang, Chengqi Zhang and Xindong Wu
Knowledge Discovery in Multiple Databases 1-85233-703-6

Jason T.L. Wang, Mohammed J. Zaki, Hannu T.T. Toivonen and Dennis Shasha (Eds)
Data Mining in Bioinformatics 1-85233-671-4

C.C. Ko, Ben M. Chen and Jianping Chen
Creating Web-based Laboratories 1-85233-837-7

Manuel Graña, Richard Duro, Alicia d'Anjou and Paul P. Wang (Eds)
Information Processing with Evolutionary Algorithms 1-85233-886-0

Colin Fyfe
Hebbian Learning and Negative Feedback Networks 1-85233-883-0

Yun-Heh Chen-Burger and Dave Robertson
Automating Business Modelling 1-85233-835-0

Dirk Husmeier, Richard Dybowski and Stephen Roberts (Eds)
Probabilistic Modeling in Bioinformatics and Medical Informatics 1-85233-778-8

Ajith Abraham, Lakhmi Jain and Robert Goldberg (Eds)
Evolutionary Multiobjective Optimization 1-85233-787-7

K.C. Tan, E.F.Khor and T.H. Lee
Multiobjective Evolutionary Algorithms and Applications 1-85233-836-9

Nikhil R. Pal and Lakhmi Jain (Eds)
Advanced Techniques in Knowledge Discovery and Data Mining 1-85233-867-9

Amit Konar and Lakhmi Jain
Cognitive Engineering 1-85233-975-6

Miroslav Kárný (Ed.)
Optimized Bayesian Dynamic Advising 1-85233-928-4

Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N. Papadopoulos and Yannis Theodoridis
R-trees: Theory and Applications 1-85233-977-2

Sanghamitra Bandyopadhyay, Ujjwal Maulik, Lawrence B. Holder and Diane J. Cook (Eds)
Advanced Methods for Knowledge Discovery from Complex Data 1-85233-989-6

Marcus A. Maloof (Ed.)
Machine Learning and Data Mining for Computer Security 1-84628-029-X

Sifeng Liu and Yi Lin
Grey Information 1-85233-995-0

Vasile Palade, Cosmin Danut Bocaniala and Lakhmi Jain (Eds)
Computational Intelligence in Fault Diagnosis 1-84628-343-4

Mitra Basu and Tin Kam Ho (Eds)
Data Complexity in Pattern Recognition 1-84628-171-7

Samuel Pierre (Ed.)
E-learning Networked Environments and Architectures 1-84628-351-5

Arno Scharl and Klaus Tochtermann (Eds)
The Geospatial Web 1-84628-826-5

Ngoc Thanh Nguyen
Advanced Methods for Inconsistent Knowledge Management 1-84628-888-3

Mikhail Prokopenko (Ed.)

Advances in Applied Self-organizing Systems

Springer

Mikhail Prokopenko, PhD, MA, MSc (Hon)
Commonwealth Scientific and Industrial Research Organisation (CSIRO)
Australia

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2007934751

AI&KP ISSN 1610-3947
ISBN: 978-1-84628-981-1 e-ISBN: 978-1-84628-982-8

Printed on acid-free paper.

9 8 7 6 5 4 3 2 1

springer.com

Preface

It was 60 years ago that a system was first termed "self-organizing" in modern scientific literature.[1] Since then, the concept of self-organization has developed in many directions and affected diverse fields, ranging from biology to physics to social sciences. For example, in his seminal book *At Home in the Universe*, Stuart Kauffman argued that natural selection and self-organization are two complementary forces necessary for evolution: "If biologists have ignored self-organization, it is not because self-ordering is not pervasive and profound. It is because we biologists have yet to understand how to think about systems governed simultaneously by two sources of order ... if ever we are to attain a final theory in biology, we will surely, surely have to understand the commingling of self-organization and selection."[2] A similar dilemma can be rephrased for various fields of engineering: If engineers have ignored self-organization, it is not because self-ordering is not pervasive and profound. It is because we engineers have yet to understand how to think about systems governed simultaneously by two sources of order: traditional design and self-organization.

Without claiming an undue comprehensiveness, this book presents state-of-the-practice of self-organizing systems and suggests a high-level breakdown of applications into two general areas:

- Distributed management and control
- Self-organizing computation

Each of these areas is exemplified with a selection of invited contributions, written and peer-reviewed by international experts in their respective fields, convincingly demonstrating achievements of self-organizing systems. The overall selection balances many aspects: modelling vs. simulation vs. deployment, as well as macro- vs. microscale.

We begin with more established fields of traffic management, sensor networks, and structural health monitoring, building up towards robotic teams, solving challenging tasks and deployed in tough environments. These scenarios mostly belong to

[1] Ashby, W. R. (1947). Principles of the Self-Organizing Dynamic System. *Journal of General Psychology*, 37:125–128.

[2] Kauffman, S. (1995). *At Home in the Universe*, p. 112. Oxford University Press, New York.

macro-level, where multiple agents (e.g., robots) themselves may contain complicated components. Nevertheless, the main topic is self-organization within a multi-agent system, brought about by interactions among the agents. The second half of the book follows with a deeper look into the microlevel, and considers local interactions among agents such as particles, cells, and neurons. These interactions lead towards self-organizing resource management, scheduling, and visualization, as well as self-modifying digital circuitry, immunocomputing, reaction-diffusion computation, and eventually to artificial life.

We believe that the broad range of scales at which self-organizing systems are applied to real-world problems is one of the most convincing arguments for acceptance of the unifying theme—practical relevance and applicability of self-organization.

Sydney, April 2007 *Mikhail Prokopenko*

Contents

Part I Introduction

1 Design vs. Self-organization
Mikhail Prokopenko 3

2 Foundations and Formalizations of Self-organization
Daniel Polani 19

Part II Distributed Management and Control

3 Self-Organizing Traffic Lights: A Realistic Simulation
Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe 41

4 A Self-organizing Sensing System for Structural Health Monitoring of Aerospace Vehicles
N. Hoschke, C. J. Lewis, D. C. Price, D. A. Scott, V. Gerasimov, and P. Wang . . . 51

5 Decentralized Decision Making for Multiagent Systems
George Mathews, Hugh Durrant-Whyte, and Mikhail Prokopenko 77

6 Learning Mutation Strategies for Evolution and Adaptation of a Simulated Snakebot
Ivan Tanev 105

7 Self-Organization as Phase Transition in Decentralized Groups of Robots: A Study Based on Boltzmann Entropy
Gianluca Baldassarre 127

8 Distributed Control of Microscopic Robots in Biomedical Applications
Tad Hogg 147

Part III Self-Organizing Computation

9 Self-Organizing Digital Systems
Nicholas J. Macias and Lisa J. K. Durbeck 177

10 Self-organizing Nomadic Services in Grids
Tino Schlegel and Ryszard Kowalczyk 217

11 Immune System Support for Scheduling
Young Choon Lee and Albert Y. Zomaya 247

12 Formal Immune Networks: Self-Organization and Real-World Applications
Alexander O. Tarakanov 271

13 A Model for Self-Organizing Data Visualization Using Decentralized Multiagent Systems
Andrew Vande Moere 291

14 Emergence of Traveling Localizations in Mutualistic-Excitation Media
Andrew Adamatzky 325

Part IV Discussion

15 A Turing Test for Emergence
Fabio Boschetti and Randall Gray 349

Index 365

Contributors

Andrew Adamatzky
University of the West of England
Bristol BS16 1QY
United Kingdom
andrew.adamatzky@uwe.ac.uk

Gianluca Baldassarre
Laboratory of Autonomous Robotics and Artificial Life
Istituto di Scienze e Tecnologie della Cognizione,
Consiglio Nazionale delle Ricerche (LARAL-ISTC-CNR)
Italy
gianluca.baldassarre@istc.cnr.it

Fabio Boschetti
Marine and Atmospheric Research
Commonwealth Scientific and Industrial Research Organisation (CSIRO)
Private Bag 5, Wembley WA 6913
Australia
fabio.boschetti@csiro.au

Seung-Bae Cools
Centrum Leo Apostel
Vrije Universiteit Brussel
Krijgskundestraat 33
B-1160 Brussel
Belgium
secools@vub.ac.be

Bart D'Hooghe
Centrum Leo Apostel
Vrije Universiteit Brussel
Krijgskundestraat 33
B-1160 Brussel
Belgium
bdhooghe@vub.ac.be

Lisa J. K. Durbeck
Cell Matrix Corporation
PO Box 510485, Salt Lake City
UT 84151
United States
ld@cellmatrix.com

Hugh Durrant-Whyte
ARC Centre of Excellence for Autonomous Systems
The University of Sydney, Sydney, NSW
Australia
hugh@acfr.usyd.edu.au

Vadim Gerasimov
Autonomous Systems Laboratory
ICT Centre
Commonwealth Scientific and Industrial Research Organisation (CSIRO)
Locked Bag 17, North Ryde, NSW 1670
Australia
vadim.gerasimov@csiro.au

Carlos Gershenson
Centrum Leo Apostel
Vrije Universiteit Brussel
Krijgskundestraat 33
B-1160 Brussel
Belgium
cgershen@vub.ac.be

Randall Gray
Marine and Atmospheric Research
Commonwealth Scientific and Industrial
Research Organisation (CSIRO)
GPO Box 1538, Hobart, TAS, 7001
randall.gray@csiro.au

Tad Hogg
Hewlett-Packard Laboratories
Palo Alto, CA
United States
tad.hogg@hp.com

Nigel Hoschke
Materials Science and Engineering
Commonwealth Scientific and Industrial
Research Organisation (CSIRO)
PO Box 218, Lindfield NSW 2070
Australia
nigel.hoschke@csiro.au

Ryszard Kowalczyk
Swinburne Centre for Information
Technology Research
Faculty of Information and
Communication Technologies
Swinburne University of Technology
Australia
rkowalczyk@ict.swin.edu.au

Young Choon Lee
School of Information Technologies
University of Sydney, Sydney, NSW
Australia
yclee@it.usyd.edu.au

Chris J. Lewis
Materials Science and Engineering
Commonwealth Scientific and Industrial
Research Organisation (CSIRO)
PO Box 218, Lindfield NSW 2070
Australia
chris.lewis@csiro.au

Nicholas J. Macias
Cell Matrix Corporation
PO Box 510485, Salt Lake City
UT 84151
United States
nmacias@cellmatrix.com

George Mathews
ARC Centre of Excellence for
Autonomous Systems
University of Sydney, Sydney, NSW
Australia
g.mathews@cas.edu.au

Daniel Polani
Adaptive Systems Research Group
Department of Computer Science
University of Hertfordshire
United Kingdom
d.polani@herts.ac.uk

Don C. Price
Materials Science and Engineering
Commonwealth Scientific and Industrial
Research Organisation (CSIRO)
PO Box 218, Lindfield NSW 2070
Australia
don.price@csiro.au

Mikhail Prokopenko
Autonomous Systems Laboratory
ICT Centre
Commonwealth Scientific and Industrial
Research Organisation (CSIRO)
Locked Bag 17, North Ryde, NSW 1670
Australia
mikhail.prokopenko@csiro.au

Tino Schlegel
Swinburne Centre for Information Technology Research
Faculty of Information and Communication Technologies
Swinburne University of Technology
Australia
tschlegel@ict.swin.edu.au

D. Andrew Scott
Materials Science and Engineering
Commonwealth Scientific and Industrial Research Organisation (CSIRO)
PO Box 218, Lindfield NSW 2070
Australia
andrew.scott@csiro.au

Ivan Tanev
Doshisha University
Department of Information Systems Design
1-3 Miyakodani, Tatara, Kyotanabe
Kyoto 610-0321
Japan
itanev@mail.doshisha.ac.jp

Alexander O. Tarakanov
St. Petersburg Institute for Informatics and Automation
Russian Academy of Sciences
14-line 39, St. Petersburg, 199178
Russia
tar@iias.spb.su

Andrew Vande Moere
Key Centre of Design Computing and Cognition
University of Sydney, Sydney, NSW
Australia
andrew@arch.usyd.edu.au

Peter Wang
Autonomous Systems Laboratory
ICT Centre
Commonwealth Scientific and Industrial Research Organisation (CSIRO)
Locked Bag 17, North Ryde, NSW 1670
Australia
peter.wang@csiro.au

Albert Y. Zomaya
School of Information Technologies
University of Sydney, Sydney, NSW
Australia
zomaya@it.usyd.edu.au

Part I

Introduction

1

Design vs. Self-organization

Mikhail Prokopenko

1.1 Introduction

The theory of self-organization has sufficiently matured over the last decades and is beginning to find practical applications in many fields. Rather than analyzing and comparing underlying definitions of self-organization—a task complicated by a multiplicity of complementary approaches in the literature (see, e.g., recent reviews by Boschetti et al. 2005; Prokopenko et al. 2007)—we investigate a possible design space for self-organizing systems and examine ways to balance design and self-organization in the context of applications.

Typically, self-organization is defined as the evolution of a system into an organized form in the absence of external pressures. A broad definition of self-organization is given by Haken: "a system is self-organizing if it acquires a spatial, temporal or functional structure without specific interference from the outside. By 'specific' we mean that the structure or functioning is not impressed on the system, but rather that the system is acted upon from the outside in a nonspecific fashion. For instance, the fluid which forms hexagons is heated from below in an entirely uniform fashion, and it acquires its specific structure by self-organization" (Haken 1988).

Another definition is offered by Camazine et al. in the context of pattern formation in biological systems: "Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern" (Camazine et al. 2001).

In our view, these definitions capture three important aspects of self-organization. Firstly, it is assumed that the system has many interacting components (agents) and advances from a less organized state to a more organized state dynamically, over some time, while exchanging energy, matter, and/or information with the environment. Secondly, this organization is manifested via global coordination, and the global behaviour of the system is a result of the interactions among the agents. In other words, the global pattern is not imposed upon the system by an external ordering influence (Bonabeau et al. 1997). Finally, the components, whose properties and behaviours are defined

prior to the organization itself, have only local information, and do not have knowledge of the global state of the system; therefore, the process of self-organization involves some local information transfer (Polani 2003).

Self-organization within a system brings about several attractive properties, in particular, robustness, adaptability and scalability. In the face of perturbations caused by adverse external factors or internal component failures, a *robust* self-organizing system continues to function. Moreover, an *adaptive* system may reconfigure when required, degrading in performance "gracefully" rather than catastrophically. In certain circumstances, a system may need to be extended with new components and/or new connections among existing modules. Without self-organization such *scaling* must be preoptimized in advance, overloading the traditional design process.

It is interesting at this stage to contrast traditional engineering methods with biological systems that evolve instead of being built by attaching separately predesigned parts together. Each biological component is reliant on other components and co-evolves to work even more closely with the whole. The result is a dynamic system where components can be reused for other purposes and take on multiple roles (Miller et al. 2000), increasing robustness observed on different levels: from a cell to an organism to an ant colony. Complementarity of coevolving components is only one aspect, however. As noted by Woese (2004), "Machines are stable and accurate because they are designed and built to be so. The stability of an organism lies in resilience, the homeostatic capacity to reestablish itself." While traditionally engineered systems may still result in brittle designs incapable of adapting to new situations, "organisms are resilient patterns in a turbulent flow—patterns in an energy flow" (Woese 2004). It is precisely this homeostatic resilience that can be captured by self-organization.

However, in general, self-organization is a not a force that can be applied very naturally during a design process. In fact, one may argue that the notions of design and self-organization are contradictory: the former approach often assumes a methodical step-by-step planning process with predictable outcomes, whereas the latter involves nondeterministic spontaneous dynamics with emergent features.

Thus, the main challenge faced by designers of self-organizing systems is how to achieve and control the desired dynamics. Erring on the one side may result in overengineering the system, completely eliminating emergent patterns and suppressing an increase in internal organization with outside influence. Strongly favouring the other side may leave too much nondeterminism in the system's behaviour, making its verification and validation almost impossible. The balance between design and self-organization is our main theme, and we hope to identify essential causes behind successful applications and propose guiding principles for future scenarios.

1.2 Background

Self-organization occurs in both biological and nonbiological systems, ranging from physics and chemistry to sociology. In nonbiological systems, it is produced by a flow of energy into or out of the system that pushes it beyond equilibrium: the winds that produce characteristic ripples in sand, the temperature gradients that produce Bénard

convection cells in a viscous fluid, the thermodynamic forces that lead to crystal growth, and characteristic molecular conformations are all examples of these external energy inputs. However, the nature of the outcomes depends critically on the interactions between the low-level components of the systems—the grains of sand, the molecules in the fluid, the atoms in the crystals and molecules—and these interactions are determined by the laws of nature and are immutable (Prokopenko et al. 2006c).

In biological systems, on the other hand, the interactions among components of a system may change over generations as a result of evolution. There are selection pressures shaping adaptation of the system (a biological organism) to the environment. These selection pressures lead to self-organization that is desirable for the survival of the system in the environment in which it has evolved, but which may be undesirable in other environments. Similarly, when using evolutionary methods for the design of applied self-organizing systems, there is a need to identify appropriate selection pressures (described more systematically in Section 1.3). These pressures constrain and channel components' interactions to produce desirable responses (Prokopenko et al. 2006c).

Self-organization is typically (but not necessarily) accompanied by the emergence of new patterns and structures. An important distinction between two kinds of emergence is identified by Crutchfield (1994):

- Pattern formation, referring to an external observer who is able to recognize how unexpected features (patterns) "emerge" during a process (e.g., spiral waves in oscillating chemical reactions). These patterns may not have specific meaning within the system, but take on a special meaning to the observer when detected.
- Intrinsic emergence, referring to the emergent features which are important within the system because they confer additional functionality to the system itself, e.g., support for global coordination and computation: for example, the emergence of coordinated behaviour in a flock of birds allows efficient global information processing through local interactions, which benefits individual agents.

In turn, the functional patterns emerging intrinsically can be further distinguished in terms of their usage: one may consider an *exploitation* of the patterns while the system is near an equilibrium or an *exploration* of patterns during the system's shift away from an equilibrium. Examples of exploration include autocatalytic processes leading to optimal paths' emergence, self-organized criticality phenomena, self-regulatory behaviour during coevolution, etc., whereas exploitation may be used during traversing of the optimal paths, self-assembly along optimal gradients, replication according to error-correcting encodings, and so on.

Self-organization has been the topic of many theoretical investigations, but its practical applications have been somewhat neglected. On the other hand, there are many studies reporting various advances in the field, and the lack of a common design methodology for these applications across multiple scales indicates a clear gap in the literature. The following short review pinpoints relevant works which, nevertheless, may set the scene for our effort.

Zambonelli and Rana (2005) discussed a variety of novel distributed computing scenarios enabled by recent advances in microelectronics, communication, and

information technologies. The scenarios are motivated by a number of challenges which, on the one hand, make it impossible for application components to rely on a priori information about their execution context, and on the other hand, make it very difficult for engineers to enforce strict microlevel control over the components. These challenges call for novel approaches to distributed systems engineering, and a point is made that the industry has also realized the importance of self-organization and decentralized management approaches (the Autonomic Computing program at IBM Research, the Dynamic Systems Initiative at Microsoft, and the Adaptive Enterprise strategy from HP). Zambonelli and Rana (2005) conclude that, "perhaps one of the barriers for real-world adoption is the lack of support in existing distributed systems infrastructure to enable these techniques to be utilized effectively." One of the aims of our effort is to explore directions towards a better adoption of self-organization as a concept for engineering distributed systems. In addition, we intend to consider several novel applications, extending the range of applicability of self-organizing systems.

Sahin and Spears (2004) consider swarm robotics—the study of how a swarm of relatively simple physically embodied agents can be constructed to collectively accomplish tasks that are beyond the capabilities of a single agent. Unlike other studies on multirobot systems, swarm robotics emphasizes self-organization and emergence, while keeping in mind the issues of scalability and robustness. These emphases promote the use of relatively simple robots, equipped with localized sensing ability, scalable communication mechanisms, and the exploration of decentralized control strategies. While definitely very valuable in addressing the task in point, this work does not expand into related areas (which are out of its scope), thus leaving interscale relationships indistinct.

Self-organizing computation is another example of an emerging application domain. Czap et al. (2005) argue that since self-organization and adaptation are concepts stemming from nature, conventional self-organization and adaptation principles and approaches are prevented from being directly applicable to computing and communication systems, which are basically artificial systems. Their book discusses these challenges, as well as a range of state-of-the-art methodologies and technologies for the newly emerging area of Self-Organization and Autonomic Informatics. What may be lacking, however, is a well-grounded connection to other application areas and identification of possible overlaps.

In summary, self-organization is a multifaceted phenomenon, present in many fields, operating at multiple scales, and performing diverse roles. We hope that the practical case studies described in this book may not only illustrate the richness of the topic, but also provide guidance to this intricate area.

1.3 Evolutionary Design

One way to address the "design vs. self-organization" dilemma is to consider possible parameters that guide the design of a self-organizing system. Let us begin with a quotation from a topical review by Scaruffi (2003), who considered the task of a "design without a designer":

> The physicist Sadi Carnot, one of the founding fathers of Thermodynamics, realized that the statistical behavior of a complex system can be predicted if its parts were all identical and their interactions weak. At the beginning of the century, another French physicist, Henri Poincaré, realizing that the behavior of a complex system can become unpredictable if it consists of few parts that interact strongly, invented "chaos" theory. A system is said to exhibit the property of chaos if a slight change in the initial conditions results in large-scale differences in the result. Later, Bernard Derrida will show that a system goes through a transition from order to chaos if the strength of the interactions among its parts is gradually increased. But then very "disordered" systems spontaneously "crystallize" into a higher degree of order.

An important lesson here is that there are transitions separating ordered and chaotic regimes, and by varying *control parameters* (e.g., the system composition and the strength of interactions within it) one may trigger these transitions. This observation by itself is not sufficient to identify the generic design space. However, several approaches, also reviewed by Scaruffi, further develop this idea. In particular, *synergetics*—a theory of pattern formation in complex systems, developed by Haken (1983b)—is relevant. Following the Ginzburg-Landau theory, Haken introduced *order parameters* in explaining structures that spontaneously self-organize in nature. When energy or matter flows into a system typically describable by many variables, it may move far from equilibrium, approach a threshold (that can be defined in terms of some control parameters), and undergo a phase transition. At this stage, the behaviour of the overall system can be described by only a few order parameters (degrees of freedom) that characterize newly formed patterns. In other words, the system becomes low-dimensional as some dominant variables "enslave" others, causing the whole system to act in synchrony. A canonical example is laser: a beam of coherent light created out of the chaotic movement of particles.

The "enslaving principle" generalizes the order parameter concept: in the vicinity of phase transitions, a few *slower* and long-lasting components of the system determine the macroscopic dynamics, whereas the *faster* and short-lasting components quickly relax to their stationary states (Jirsa et al. 2002). The fast-relaxing components represent stable modes, e.g., the chaotic motion of particles. The slower components represent unstable modes, i.e., the coherent macroscopic structure and behaviour of the whole system (see also this volume Chapter 2). Thus, the order parameters can be interpreted as the amplitudes of these unstable modes that determine the macroscopic pattern and the dynamics of the enslaved fast-relaxing modes. In particular, the stationary states of fast-relaxing components are determined by the order parameters.

It can be argued that a layered hierarchical structure emerges where "higher" levels (the order parameters) "control" or "force order upon" lower levels (short-lasting and fast-relaxing components) (Liljenström and Svedin 2005). However, we may also point out the circular nature of the mechanism: the dynamics of microscopic short-lasting components brings the system to the phase transition, forcing it over a threshold and stimulating macroscopic pattern formation. When new macroscopic patterns emerge, the order parameters enforce the downward enslavement (Haken 1983a):

> Because the order parameter forces the individual electrons to vibrate exactly in phase, thus imprinting their actions on them, we speak of their "enslavement" by the order parameter. Conversely, these very electrons generate the light wave, i.e., the order parameter, by their uniform vibration.

The collective synchronization of oscillators is another example of such circular causality. It is well known that coupled limit-cycle oscillators tend to synchronize by altering their frequencies and phase angles (Kuramoto 1984; Pikovsky et al. 2001). Given a pulse, initially a few oscillators become synchronized, then a mean field forms that drives other oscillators which, in turn, contribute to the mean field. Thus, such self-organization can be better characterized in terms of tangled hierarchies exhibiting the Strange Loops described by Hofstadter: "an interaction between levels in which the top level reaches back down towards the bottom level and influences it, while at the same time being itself determined by the bottom level" (Hofstadter 1989).

Importantly, tangled hierarchies with circular causality between microscopic and macroscopic levels result in stable behaviour. For example, as noted by M.J.M. Volman (1997) in the context of neurobehavioural studies:

> Enslaving provides a parsimonious solution to the degrees of freedom problem: only one or a few variables have to be controlled by the central nervous system. Two variables play an essential role: the order parameter or collective variable, and the control parameter. The collective variable captures the intrinsic order of the system. The control parameter is the parameter that induces a phase transition from one stable state of the system to another.

A self-organized low-dimensional system with fewer available configurations may be more efficient than a high-dimensional disorganized system which may, in principle, access more configurations. The reason for such higher efficiency is explained by Kauffman (2000), who suggested that the underlying principle of self-organization is the generation of constraints in the release of energy. According to this view, the constrained release allows for such energy to be controlled and channelled to perform some useful work. This work is "propagatable" and can be used in turn to create better and more efficient constraints, releasing further energy, and so on. Importantly, the ability to constrain and control the release of energy provides the self-organized system with a variety of behaviours that can be selectively chosen for successful adaptation (Prokopenko et al. 2007), thus conforming with Ashby's law of requisite variety.

These observations further advance our search for a suitable design space: control parameters become optimization variables, while order parameters contribute to (multi)objective functions. The overall optimization is to be solved under the constraints generated by the release of energy from components in the system. The three elements (variables, objective functions, and constraints) constitute the design space. This approach suggests considering *evolutionary design* as the methodology for designing self-organizing systems. Typically, evolutionary design may employ genetic algorithms in evolving optimal strategies that satisfy given fitness functions by exploring large and sophisticated search-space landscapes (Crutchfield et al. 1998; Miller et al. 2000). With selection and genetic variation of microscopic components, evolu-

tionary design is capable of discovering macroscopic patterns that correspond to order parameters.

There is a fundamental reason for employing evolutionary methods in designing self-organizing nonlinear systems. As pointed out by Heylighen (2000), many intricacies associated with nonlinearity (e.g., limit cycles, chaos, sensitivity to initial conditions, dissipative structures) can be interpreted through the interplay of positive and negative feedback cycles. In turn, both types of feedback provide a selective advantage: when variations positively reinforce themselves (e.g., autocatalytic growth) the number and diversity of configurations are increased to the point where resources may become insufficient, and competition may intensify. On the other hand, when variations reduce themselves via negative feedback, configurations become more stable. Therefore, a self-organizing system with both types of feedback is a natural target for evolutionary design.

Heylighen further notes that the increase in organization can be measured quantitatively as a decrease in statistical entropy, exported by the self-organizing system into its surroundings. Prigogine called systems which continuously export entropy in order to maintain their organization *dissipative structures* (Prigogine 1980), and formulated the minimum entropy production principle: stable near-equilibrium dissipative systems minimize their rate of entropy production. While the practical applicability of this principle is still a subject of ongoing debate, we believe that it identifies a generic guiding rule for evolutionary design, suggesting the incorporation of minimization of entropy rate in the employed fitness functions.

Consequently, we may approach evolutionary design in two ways: via task-specific objectives or via generic intrinsic selection criteria (Prokopenko et al. 2006a,b). The latter method—*information-driven evolutionary design*—essentially focuses on information transfer within specific channels, enabling "propagatable" work, i.e., self-organization. Various generic information-theoretic criteria may be considered, e.g.:

- Maximization of information transfer in perception-action loops (Klyubin et al. 2004, 2005).
- Minimization of heterogeneity in agent states, measured with the variance of the rule-space's entropy (Wuensche 1999; Prokopenko et al. 2005d) or Boltzmann entropy in agent states (Baldassarre et al. 2007); see also this volume Chapter 7.
- Stability of multiagent hierarchies (Prokopenko et al. 2005d).
- Efficiency of computation (computational complexity).
- Efficiency of communication topologies (Prokopenko et al. 2005b,c); see also this volume Chapter 4.
- Efficiency of locomotion and coordination of distributed actuators (Der et al. 1999; Tanev et al. 2005; Prokopenko et al. 2006a,b); see also Chapter 6.[1]

The solutions obtained by information-driven evolution can be judged by their degree of approximation of direct evolutionary computation, where the latter uses task-specific objectives. A good approximation indicates that the chosen criteria capture information dynamics of self-organization within specific channels.

[1] Chapter numbers refer to other chapters within this volume.

In summary, design is possible even when the target is a far-from-equilibrium nonlinear system with multiple independent and interacting units. One should not avoid far-from-equilibrium dynamics and symmetry-breaking behaviour, but rather exploit the opportunities for creating stable patterns out of fluctuations.[2] The regions of macroscopic stability correspond to order parameters. These regions are separated by phase transitions that can be quantified via entropy rate and induced by varying the control parameters. In short, one should design local rules of interaction among microscopic components (including the constraints and control variables) in such a way that macroscopic patterns (measured via the objective functions) self-organize globally, being then selected by information-driven evolution.

Example: Self-organizing Locomotion

Different internal channels through which information flows within the system may be chosen for a specific analysis. For example, let us consider a modular robotic system modelling a multisegment snakelike (salamander) organism, with actuators ("muscles") attached to individual segments ("vertebrae"). A particular side-winding locomotion emerges as a result of individual control actions when the actuators are coupled within the system and follow specific evolved rules, as described in Chapter 6 as well as by Tanev et al. (2005). There is no global coordinating component in the evolved system, and it can be shown that as the modular robot starts to move across the terrain, the distributed actuators become more coupled when a coordinated side-winding locomotion is dominant. The periodicity of the side-winding locomotion can be related to order parameter(s). Faced with obstacles, the robot temporarily loses the side-winding pattern: the modules become less organized, the strength of their coupling (which can be selected as a control parameter) is decreased, and rather than exploiting the dominant pattern, the robot explores various alternatives. Such exploration temporarily decreases self-organization within the system. When the obstacles are avoided, the modules "rediscover" the dominant side-winding pattern by themselves, manifesting again the ability to self-organize without any global controller. Of course, the "magic" of this self-organization is explained by properties defined a priori: the control rules employed by the biologically inspired actuators have been obtained by a genetic programming algorithm, while the biological counterpart (the rattlesnake *Crotalus cerastes*) naturally evolved over a long period of time. Our point is simply that these transitions can be quantitatively measured within the channels of interest (e.g., via generalized entropy rate and excess entropy) and used in information-driven evolutionary design (Prokopenko et al. 2006a,b).

1.4 Information Dynamics

The generation of constraints in the release of energy explains why a low-dimensional self-organized system with fewer available configurations is more efficient than a

[2] According to Prigogine (1980), a thermodynamic system can be in a steady state while being not in equilibrium.

high-dimensional disorganized system. One quantitative interpretation of this is that many actual configurations of a disorganized system may not be statistically different (Prokopenko et al. 2007). On the other hand, as the system self-organizes, it reaches a progressively larger number of statistically different configurations. In other words, an increase in organization can be measured via an increase in statistical complexity of the system's dynamics (Crutchfield 1994; Shalizi 2001; Shalizi et al. 2004). The latter approach is formalized within the computational mechanics methodology, which equates statistically different configurations with *causal states*.[3]

Recently, Correia (2006) analyzed self-organization motivated by embodied systems, i.e., physical systems situated in the real world, and established four fundamental properties of self-organization: no external control, an increase in order, robustness,[4] and interaction. All of these properties are easily interpretable in terms of information dynamics (Prokopenko et al. 2007). Firstly, the absence of external control may correspond to spontaneous information transfer within the system without any flow of information into the self-organizing system. Secondly, an increase in order or complexity reflects simply that the statistical complexity is increased internally within the system: $C_\mu^{\mathrm{system}}(t_2) > C_\mu^{\mathrm{system}}(t_1)$, for $t_2 > t_1$, where $C_\mu^{\mathrm{system}}(t)$ is the statistical complexity at time t. In general, the distinction between these two requirements may be relaxed (Prokopenko et al. 2007), resulting in the requirement that in a self-organizing system the complexity of external influence $C_\mu^{\mathrm{influence}}$ is strictly less than the gain in internal complexity, $\triangle C_\mu^{\mathrm{system}} = C_\mu^{\mathrm{system}}(t_2) - C_\mu^{\mathrm{system}}(t_1)$, within the system:

$$C_\mu^{\mathrm{influence}} < \triangle C_\mu^{\mathrm{system}}$$

Thirdly, a system is robust if it continues to function in the face of perturbations (Wagner 2005)—in terms of information dynamics, robustness of a self-organizing system to perturbations means that it may interleave stages of an increased information transfer within some channels (dominant patterns are being exploited) with periods of decreased information transfer (alternative patterns are being explored). Finally, the interaction property is described by Correia (2006) as follows: minimization of local conflicts produces global optimal self-organization, which is evolutionarily stable. Following the review by Prokopenko et al. (2007), we note that minimization of local conflicts corresponds to a reduction of assortative noise (or nonassortativeness within the system, thus increasing the information transfer within the system.

Example: Self-organizing Traffic

In the context of pedestrian traffic, Correia (2006) argues that it can be shown that the "global efficiency of opposite pedestrian traffic is maximized when interaction rate is locally minimized for each component. When this happens two separate lanes form, one in each direction. The minimization of interactions follows directly from maximizing the average velocity in the desired direction." In other words, the division into

[3] Statistical complexity is also an upper bound of predictive information, or structure, within the system (Bialek et al. 2001; De Wolf and Holvoet 2005).

[4] Although Correia refers to this as adaptability, he in fact defines robustness.

lanes results from maximizing velocity (an overall objective or fitness), which in turn supports minimization of conflicts. A practical case study of self-organizing traffic, presented in Chapter 3, considers ways to minimize conflicts as well, e.g., via "platoons" or "convoys" of cars that move together, improving the traffic flow.

Another example is provided by ants: "Food transport is done via a trail, which is an organized behaviour with a certain complexity. Nevertheless, a small percentage of ants keep exploring the surroundings and if a new food source is discovered a new trail is established, thereby dividing the workers by the trails (Hubbell et al. 1980) and increasing complexity" (Correia 2006). Here, the division into trails is again related to an increase in fitness and complexity.

These examples demonstrate that when local conflicts are minimized, the degree of coupling among the components (i.e., interaction) increases and the information flows more easily, thus increasing the information transfer. This means that self-organization as a dynamic process tends to increase the overall diversity of a system (more lanes or trails), while keeping the interplay among different channels (the assortative noise within the system, the conflicts) in check. In summary, self-organization can be quantitatively studied via information dynamics when the appropriate channels are identified (Prokopenko et al. 2007).

Example: Self-organizing Computation

In illustrating the phenomenon of self-organizing computation we employ Cellular Automata (CA)—discrete spatially extended dynamical systems that are often used as models of computational, physical, and biological processes—see, e.g., (Mitchell et al. 1993) and Chapter 14. The main conjecture within this application domain is that physical systems achieve the prerequisites for computation (i.e., transmission, storage, modification) in the vicinity of a phase transition between periodic and chaotic behaviour (Langton 1991).

In classifying CA rules according to their asymptotic behaviour, the following qualitative taxonomy is typically employed: class I (homogeneity), class II (periodicity), class III (chaos), and class IV (complexity) (Wolfram 1984). The first class consists of CA that, after a finite number of time steps, produce a unique, homogeneous state (analogous to "fixed points" in phase space). From almost all initial states, such behaviour completely destroys any information on the initial state, i.e., complete prediction is trivial and complexity is low. The second class contains automata which generate a set of either stable or periodic structures (typically having small periods—analogous to "limit cycles" in phase space); each region of the final state depending only on a finite region of the initial state; i.e., information contained within a small region in the initial state suffices to predict the form of a region in the final state. The third class includes infinite CA producing aperiodic ("chaotic") spatiotemporal patterns from almost all possible initial states; the effects of changes in the initial state almost always propagate forever at a finite speed, and a particular region depends on a region of the initial state of an ever-increasing size (analogous to "chaotic attractors" in phase space). While any prediction of the "final" state requires complete knowledge of the initial state, the regions are indistinguishable statistically as they possess

no structure, and therefore the statistical complexity is low. The fourth class deals with automata that generate patterns continuously changing over an unbounded transient.

The fourth class CA existing near the phase transition between periodic and chaotic behaviour was shown to be capable of universal computation (Wolfram 1984). These CA support three basic operations (information storage, transmission, and modification) through static, propagating, and interacting structures (blinkers, gliders, collisions). Importantly, the patterns produced along the transient are different in terms of generated structure, and, in fact, their structural *variability* is the highest of all the four classes—i.e., the information transfer and the complexity of the class IV automata are the highest. Casti (1991) developed an analogy between the complex (class IV) automata and quasi-periodic orbits in phase space while pursuing deeper interconnections between CA, dynamical systems, Turing machines, and formal logic systems—in particular, the complex class IV automata were related to formal systems with undecidable statements (Gödel's theorem). These interconnections are also explored in Chapter 15, which investigates the concept of *emergence* and the limits of algorithmic approaches.

1.5 Discussion and Conclusion

Our analysis would be incomplete without a discussion of a few obstacles that prevent a straightforward application of self-organizing systems in industry. The limits of algorithmic approaches to emergence studied in Chapter 15 may manifest themselves in many different ways. Firstly, the "numerous interactions among the lower-level components" (Camazine et al. 2001) that are essential for self-organization may often be costly in terms of communication overhead. For example, many decentralized multiagent (e.g., peer-to-peer) clustering algorithms deployed in sensor networks form stable clusters only after a significant number of messages (Prokopenko et al. 2005a), potentially incurring a prohibitive cost. This highlights even more the role of selecting the most important information channels and communication topologies, reducing local conflicts (assortative noise) and maximizing information transfer.

Secondly (and this is probably the most principled impediment), self-organization results in nondeterministic outcomes. In fact, this is one of its strengths, and as noted earlier, far-from-equilibrium dynamics and symmetry-breaking behaviour may lead to stable patterns that can and should be exploited. However, in order to be adopted by industry, nondeterminism of self-organizing patterns requires an appropriate verification of the outcomes, and the search for the most suitable verification methodology is still open. For example, Chapter 5 investigates self-organizing collaboration within a multiagent system using analytical techniques, in an attempt to provide a verifiable optimal solution to a decentralized decision problem.

Finally, a complete self-organizing system would typically depart too strongly from incremental advancements accepted by an industry. A more realistic approach suggests, instead, the deployment of hybrid systems (e.g., such as the one described in Chapter 4) where self-organization is used within separate components, providing a convenient mechanism for managing communication overheads and verification

requirements. Such hybrid systems are an intermediate step on the path towards complete self-organizing solutions, e.g., a completely self-organizing computation within reaction-diffusion media, explored in Chapter 14.

In summary, we believe that information-driven evolutionary design can produce self-organizing systems that can be as reliable as traditionally engineered verifiable (provably-correct) systems, and as resilient as homeostatic biological organisms.

Acknowledgements

The author would like to thank members of the discussion group on "Entropy and Self-Organisation in Multi-Agent Systems," particularly Vadim Gerasimov, Nigel Hoschke, Joseph Lizier, George Mathews, Mahendra Piraveenan, and Don Price. The support of the CSIRO Emergence interaction task and the CSIRO Complex Systems Science Theme is gratefully acknowledged. Special thanks to Fabio Boschetti, Tony Farmer, Tomonari Furukawa, Carlos Gershenson, Geoff James, Antonio Lafusa, Ron Li, Abhaya Nayak, Daniel Polani, Geoff Poulton, Alex Ryan, Ivan Tanev, Tino Schlegel, Phil Valencia, and X. Rosalind Wang for their insightful comments.

References

Baldassarre, G., Parisi, D., and Nolfi, S. (2007). Measuring coordination as entropy decrease in groups of linked simulated robots. In Bar-Yam, Y., editor, *Proceedings of the 5th International Conference on Complex Systems (ICCS2004)*.

Bialek, W., Nemenman, I., and Tishby, N. (2001). Complexity through nonextensivity. *Physica A*, 302:89–99.

Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., and Camazine, S. (1997). Self-organisation in social insects. *Trends in Ecology and Evolution*, 12(5):188–193.

Boschetti, F., Prokopenko, M., Macreadie, I., and Grisogono, A.-M. (2005). Defining and detecting emergence in complex networks. In Khosla, R., Howlett, R. J., and Jain, L. C., editors, *Knowledge-Based Intelligent Information and Engineering Systems, 9th International Conference, KES 2005, Melbourne, Australia, September 14-16, 2005, Proceedings, Part IV*, volume 3684 of *Lecture Notes in Computer Science*, pages 573–580. Springer, Berlin.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2001). *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ.

Casti, J. L. (1991). Chaos, Gödel and Truth. In Casti, J. L. and Karlqvist, A., editors, *Beyond Belief: Randomness, Prediction, and Explanation in Science*. CRC Press, Boca Raton, Fla.

Correia, L. (2006). Self-organisation: a case for embodiment. In Gershenson, C. and Lenaerts, T., editors, *The Evolution of Complexity Workshop at Artificial Life X: Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems*, pages 111–116.

Crutchfield, J. P. (1994). The calculi of emergence: computation, dynamics, and induction. *Physica D*, 75:11–54.

Crutchfield, J. P., Mitchell, M., and Das, R. (1998). The evolutionary design of collective computation in cellular automata. Technical Report 98-09-080, Santa Fe Institute Working Paper, available at http://www.santafe.edu/projects/evca/Papers/EvDesign.html.

Czap, H., Unland, R., Branki, C., and Tianfield, H. (2005). *Self-Organization and Autonomic Informatics (I)*, volume 135 of *Frontiers in Artificial Intelligence and Applications*. IOS, Amsterdam.

De Wolf, T., and Holvoet, T. (2005). Emergence versus self-organisation: Different concepts but promising when combined. In Brueckner, S., Serugendo, G. D. M., Karageorgos, A., and Nagpal, R., editors, *Engineering Self-Organising Systems*, pages 1–15. Springer, Berlin.

Der, R., Steinmetz, U., and Pasemann, F. (1999). Homeokinesis: A new principle to back up evolution with learning. *Concurrent Systems Engineering Series*, 55:43–47.

Haken, H. (1983a). *Advanced Synergetics: Instability Hierarchies of Self-Organizing Systems and Devices*. Springer, Berlin.

Haken, H. (1983b). *Synergetics, an Introduction: Nonequilibrium Phase Transitions and Self-Organization in Physics, Chemistry, and Biology*, 3rd rev. enl. ed. Springer, New York.

Haken, H. (1988). *Information and Self-Organization: A Macroscopic Approach to Complex Systems*. Springer, Berlin.

Heylighen, F. (2000). Self-organization. In Heylighen, F., Joslyn, C., and Turchin, V., editors, *Principia Cybernetica Web*. Principia Cybernetica, Brussels, available at http://pespmc1.vub.ac.be/SELFORG.html.

Hofstadter, D. R. (1989). *Gödel, Escher, Bach: An Eternal Golden Braid*. Vintage, New York.

Hubbell, S. P., Johnson, L. K., Stanislav, E., Wilson, B., and Fowler, H. (1980). Foraging by bucket-brigade in leafcutter ants. *Biotropica*, 12(3):210–213.

Jirsa, V. K., Jantzen, K. J., Fuchs, A., and Kelso, J. A. (2002). Spatiotemporal forward solution of the EEG and MEG using network modeling. *IEEE Transactions on Medical Imaging*, 21(5):493–504.

Kauffman, S. A. (2000). *Investigations*. Oxford University Press, Oxford.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2004). Organization of the information flow in the perception-action loop of evolved agents. In *Proceedings of 2004 NASA/DoD Conference on Evolvable Hardware*, pages 177–180. IEEE Computer Society.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). All else being equal be empowered. In Capcarrère, M. S., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J., editors, *Advances in Artificial Life, 8th European Conference, ECAL 2005, Canterbury, U.K., September 5-9, 2005, Proceedings*, volume 3630 of *Lecture Notes of Computer Science*, pages 744–753. Springer, Berlin.

Kuramoto, Y. (1984). *Chemical Oscillations, Waves, and Turbulence*. Springer, Berlin.

Langton, C. (1991). Computation at the edge of chaos: Phase transitions and emergent computation. In Forest, S., editor, *Emergent Computation*. MIT Press, Cambridge, MA.

Liljenström, H., and Svedin, U. (2005). System features, dynamics, and resilience — some introductory remarks. In Liljenström, H. and Svedin, U., editors, *MICRO-MESO-MACRO: Addressing Complex Systems Couplings*, pages 1–16. World Scientific, Singapore.

Miller, J. F., Job, D., and Vassilev, V. K. (2000). Principles in the evolutionary design of digital circuits - Part I. *Journal of Genetic Programming and Evolvable Machines*, 1(1):8–35.

Mitchell, M., Hraber, P. T., and Crutchfield, J. P. (1993). Revisiting the edge of chaos: evolving cellular automata to perform computations. *Complex Systems*, 7:89–139.

Pikovsky, A., Rosenblum, M., and Kurths, J. (2001). *Synchronization: A Universal Concept in Nonlinear Science*. Cambridge University Press, Cambridge, UK.

Polani, D. (2003). Measuring self-organization via observers. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J. T., and Ziegler, J., editors, *Advances in Artificial life - Proceedings of the 7th European Conference on Artificial Life (ECAL), Dortmund*, pages 667–675. Springer, Heidelberg.

Prigogine, I. (1980). *From Being to Becoming: Time and Complexity in the Physical Sciences*. W. H. Freeman, San Francisco.

Prokopenko, M., Piraveenan, M., and Wang, P. (2005a). On convergence of dynamic cluster formation in multi-agent networks. In Capcarrère, M. S., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J., editors, *Advances in Artificial Life, 8th European Conference, ECAL 2005, Canterbury, UK, September 5-9, 2005, Proceedings*, volume 3630 of *Lecture Notes in Computer Science*, pages 884–894. Springer.

Prokopenko, M., Wang, P., Foreman, M., Valencia, P., Price, D. C., and Poulton, G. T. (2005b). On connectivity of reconfigurable impact networks in ageless aerospace vehicles. *Journal of Robotics and Autonomous Systems*, 53(1):36–58.

Prokopenko, M., Wang, P., and Price, D. C. (2005c). Complexity metrics for self-monitoring impact sensing networks. In Lohn, J., Gwaltney, D., Hornby, G., Zebulum, R., Keymeulen, D., and Stoica, A., editors, *Proceedings of 2005 NASA/DoD Conference on Evolvable Hardware (EH-05), Washington D.C., 29 June–1 July 2005*, pages 239–246. IEEE Computer Society, Los Alamitos, CA.

Prokopenko, M., Wang, P., Price, D. C., Valencia, P., Foreman, M., and Farmer, A. J. (2005d). Self-organizing hierarchies in sensor and communication networks. *Artificial Life, Special Issue on Dynamic Hierarchies*, 11(4):407–426.

Prokopenko, M., Gerasimov, V., and Tanev, I. (2006a). Evolving spatiotemporal coordination in a modular robotic system. In Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J. C. T., Marocco, D., Meyer, J.-A., Miglino, O., and Parisi, D., editors, *From Animals to Animats 9: 9th International Conference on the Simulation of Adaptive Behavior (SAB 2006), Rome, Italy, September 25-29 2006*, volume 4095 of *Lecture Notes in Computer Science*, pages 558–569.

Prokopenko, M., Gerasimov, V., and Tanev, I. (2006b). Measuring spatiotemporal coordination in a modular robotic system. In Rocha, L., Yaeger, L., Bedau, M., Floreano, D., Goldstone, R., and Vespignani, A., editors, *Artificial Life X: Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems*, pages 185–191, MIT Press, Bloomington IN.

Prokopenko, M., Poulton, G. T., Price, D. C., Wang, P., Valencia, P., Hoschke, N., Farmer, A. J., Hedley, M., Lewis, C., and Scott, D. A. (2006c). Self-organising impact sensing networks in robust aerospace vehicles. In Fulcher, J., editor, *Advances in Applied Artificial Intelligence*, pages 186–233. Idea Group, Hershey, PA.

Prokopenko, M., Boschetti, F., and Ryan, A. (2007). An information-theoretic primer on complexity, self-organisation and emergence. *Advances in Complex Systems, under review*.

Sahin, E. and Spears, W. M. (2004). *Swarm Robotics, Proccedings of SAB-2004 International Workshop, Santa Monica, CA, July 17, 2004, Revised Selected Papers*, volume 3342 of *Lecture Notes in Computer Science*.

Scaruffi, P. (2003). *Thinking About Thought: A Primer on the New Science of Mind, Towards a Unified Understanding of Mind, Life and Matter*. Writers Club Press, Lincoln, Neb.

Shalizi, C. (2001). *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Michigan, available at http://www.cscs.umich.edu/ crshalizi/thesis/.

Shalizi, C. R., Shalizi, K. L., and Haslinger, R. (2004). Quantifying self-organization with optimal predictors. *Physical Review Letters*, 93(11):118701–1–4.

Tanev, I., Ray, T., and Buller, A. (2005). Automated evolutionary design, robustness, and adaptation of sidewinding locomotion of a simulated snake-like robot. *IEEE Transactions on Robotics*, 21:632–645.

Volman, M. J. M. (1997). *Rhythmic Coordination Dynamics in Children with and without a Developmental Coordination Disorder*. PhD dissertation, University of Groningen, available at http://irs.ub.rug.nl/ppn/163776687.

Wagner, A. (2005). *Robustness and Evolvability in Living Systems*. Princeton University Press, Princeton, NJ.

Woese, C. R. (2004). A new biology for a new century. *Microbiology and Molecular Biology Reviews*, 68(2):173–186.

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D*, 10.

Wuensche, A. (1999). Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the z parameter. *Complexity*, 4(3):47–66.

Zambonelli, F., and Rana, O. F. (2005). Self-organization in distributed systems engineering. *Special Issue of IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 35(3).

2

Foundations and Formalizations of Self-organization

Daniel Polani

2.1 Introduction

In the study of complex systems, the relevance of the phenomenon of *self-organization* is ubiquitous. For example the stripe formation in morphogenesis (Meinhardt 1972, 1982), reaction-diffusion automata (Turing 1952), the reorganization of a self-organizing Kohonen map, the seemingly effortless distributed organization of work in an ant colony, the formation of flows in pedestrian movement patterns (Helbing et al. 2005), the maintenance and creation of the complexities involved in the maintenance of life in living cells all produce behaviour that, in one way or another, can be called "organized" and if the source of organization is not explicitly identified outside of the system, it can be called "*self*-organized."

Strangely enough, as much agreement as there is on visual inspection about whether self-organization is present or absent in a system, as little agreement exists concerning the precise meaning of the word. In other words, whereas the phenomenology of the phenomenon is pretty much agreed upon, its formal foundations are the subject to delete.

Among other problems, this causes a certain amount of confusion. For instance, the difference between the notions of emergence and self-organization is being strongly emphasized (Shalizi 2001), notwithstanding the frequent co-occurrence of these notions. On the other hand, without a clear formal definition of self-organization and emergence, it is difficult to make strong points in favour (or against) a separation of these notions.

With recent interest in the exploitation of aspects of self-organization for engineering and other applications, the importance of characterizing and understanding the phenomenon of self-organization has even increased. It is no longer sufficient to characterize a system in "ivory-tower" fashion to be in one group or another according to some human classification. Rather, it becomes necessary to work towards a predictable and structured theory that will also admit useful predictions about the performance of a system. The advent of nanotechnology and bio-inspired engineering architectures increases the immediate practical relevance of understanding and characterizing self-organization.

In view of the various streams and directions of the field of self-organization, it is beyond the scope of the present introductory chapter to review all the currents of research in the field. Rather, the aim of the present section is to address some of the points judged most relevant and to provide a discussion of suitable candidate formalisms for the treatment of self-organization. In the author's opinion, discussing formalisms is not just a vain exercise, but allows one to isolate the essence of the notion one wishes to develop. Thus even if one disagrees with the path taken (as is common in the case of not yet universally agreed upon formal notions), starting from operational formalisms serves as a compass to guide one towards notions suitable for one's purposes. This is the philosophy of the present chapter.

The chapter is structured as follows: in Section 2.2, we present several central conceptual issues relevant in the context of self-organization. Some historical remarks about related work are given in Section 2.3. To illustrate the setting, a brief overview of some classical examples for self-organizing processes is given in Section 2.4. Sections 2.5 and 2.6, we introduce the two main information-theoretic concepts of self-organization that the chapter aims to discuss. One concept, based on the ϵ-machine formalism by Crutchfield and Shalizi, describes self-organization as an increase in (statistical) complexity with time. The other concept suggests measuring self-organization as an increase in mutual correlations (measured by multi-information) between different components of a system. In Section 2.7, finally, important properties of these two measures as well as their distinctive characteristics (namely their power to identify temporal vs. compositional self-organization) are discussed; Section 2.8 offers some concluding remarks.

2.2 General Comments

In line with the comments above, the present chapter does not attempt to answer the question: "What is self-organization?" Instead, the question is "Where do we agree self-organization exists?" or "What are candidate characterizations of self-organization?." Rather than attempting to give ultimate answers, a number of suggestions are offered that can form a starting point for the development of the reader's own notion.

The distinction between self-organization and emergence is emphasized time and time again (Shalizi 2001); this is sometimes confusing, given that often both appear in similar contexts and that there is no universally accepted formal definition for either of them. This distinction emphasizes that there is a general desire to have them carry different meanings.

Self-organization, the main focus of the present chapter,[1] is a phenomenon under which a dynamical system exhibits the tendency to create organization "out of itself," without being driven by an external system, in particular, not in a "top-down" way. This requires clarification in regard to several questions:

[1] Emergence is briefly discussed in Sections 2.5.2 and 2.6.1.

1. What is meant by organization?
2. How and when to distinguish system and environment?
3. How can external drives be measured?
4. What does top-down mean?

Question 1 is clearly related to the organizational concept of *entropy*. However, it is surprisingly not straightforward to adapt entropy for use in measuring self-organization, and some additional effort must be made (Polani 2003). This is the question that the present chapter focuses on. All the other questions are only mentioned briefly to provide the reader with an idea of what further issues in the context of formalization of self-organization could and should be addressed in future.

Question 2 is, basically, about how one defines the boundaries of the system, which is related to the question of autonomy (Bertschinger et al. 2006). Once they are defined, we can ask ourselves all kinds of questions about the system under investigation and its environment, its "outside." A complication is brought into the discussion through the fact that if organization is produced in the "inner" system out of itself (the "self" in self-organization) in a real physical system, disorder has to be produced in the outside world, due to the conservation of phase space volume (Adami 1998).

However, for many so-called self-organizing systems the physical reality is quite detached, i.e., conservation or other thermodynamic laws are not (and need not be) part of the model dynamics, so Landauer's principles (Landauer 1961; Bennett and Landauer 1985) are irrelevant in the general scenario: for instance, a computer emulating a self-organizing map produces much more total heat in its computation than the minimum amount demanded by Landauer's principle due to the entropy reduction of the pure computation. In other words, the systems under consideration may be arbitrarily far away from thermal equilibrium, and, worse, there may be a whole hierarchy of different levels of organization whose constraints and invariants have to be respected before one can even consider coming close to the Landauer limit.[2]

Therefore, unless we are aiming for an understanding of nanosystems, where issues of Landauer's principle could begin to play a role, we can and will ignore issues of the "compulsory" entropy production of a real physical system that exhibits self-organization. In particular, the systems we consider in the following pages are general dynamical systems. We do not require them to be modelled in a thermodynamically or energetically consistent way.

The response to question 3 is not as straightforward, a difficulty that is conceded in Shalizi et al. (2004), who suggest studying causal inference as a possible formalism to investigate the influence of an environment on a given system. In fact, the concept of *information flow* has recently been introduced to address exactly this question (Ay and Wennekers 2003; Klyubin et al. 2004; Ay and Krakauer 2006; Ay and Polani 2007), providing an information-theoretic approach to measure causal inference. At this point these notions are still quite fresh and not much is known about their possible relevance for characterizing self-organizing systems, although it is an interesting avenue for future work.

[2] As an example, the energy balance of real biological computation process will operate at the ATP metabolism level and respect its restrictions—but this is still far off the Landauer limit.

Question 4 again introduces an interesting and at the same time unclear notion of "top-down." Roughly, top-down indicates a kind of downward causation (Emmeche et al. 2000), where one intervenes to influence some coarse-grained, global degrees of freedom to achieve a particular organizational outcome; or else, the external experimenter "micromanages" the system into a particular state. For this latter view, one could consider using a formalization of an agent manipulating its environment (Klyubin et al. 2004). This is again outside of the scope of the present chapter. Nevertheless, it is hoped that this section's brief discussion of general conceptual issues highlights some related open questions of interest that may prove amenable to treatment by a consistent theoretical framework.

2.3 Related Work and Historical Remarks

Shalizi et al. (2004) track the first use of the notion of "self-organizing systems" back to Ashby (1947). The bottom-up cybernetic approach of early artificial intelligence (Walter 1951; Pask 1960) devoted considerable interest and attention to the area of self-organizing systems; many of the questions and methods considered relevant today were appropriately identified almost half a century ago (e.g., Yovits and Cameron 1960).

The notions of *self-organization* and the related notion of *emergence* form the backbone for the studies of dynamical hierarchies and, in particular, those types of dynamics that lead to climbing the ladder of complexity as found in nature. Notwithstanding the importance and frequent use of these notions in the relevant literature, a mathematical definition that is both precise and useful remains elusive. While there is a high degree of intuitive consensus on what type of phenomena should be called "self-organizing" or "emergent," the prevailing strategy of characterization is along the line of "I know it when I see it" (Harvey 2000).

Specialized formal literature often does not go beyond pragmatic characterizations; e.g., Jetschke (1989) defines a system as undergoing a self-organizing transition if the symmetry group of its dynamics changes to a less symmetrical one (e.g., a subgroup of the original symmetry group; Golubitsky and Stewart 2003), typically occurring at phase transitions (Reichl 1980). This latter view relates self-organization to phase transitions. However, there are several reasons to approach the definition of self-organization in a different way. The typical complex system is not in thermodynamic equilibrium (see also Prigogine and Nicolis 1977). One possible extension of the formalism is towards nonequilibrium thermodynamics, identifying phase transitions by *order parameters*. These are quantities that characterize the "deviation" of the system in a more organized state (in the sense of Jetschke) from the system in a less organized state, measured by the absence or presence of symmetries. Order parameters have to be constructed by explicit inspection of the system since a generic approach is not available, although an ϵ-machine-based approach such as in Shalizi (2001) seems promising. Moreover, in complex systems, one cannot expect the a priori existence or absence of any symmetry to act as a universal indicator for self-organization; in general such a system will exhibit, at best, only approximate or imperfect symmetries, if any

at all. Without a well-founded concept of characterizing such imperfect "soft" symmetries, the symmetry approach to characterizing self-organization is not sufficient to characterize general complex systems.

2.4 Examples for Self-organization

We now consider a number of examples of systems which are typically regarded as self-organizing. Since the field lacks a consensus on suitable formal definitions, it is helpful to consider examples of the phenomena at hand, where there is less controversy as to whether or not they exhibit the desired behaviour.

2.4.1 Bifurcation

Consider a dynamical system with state $\mathbf{x}(t) \in \mathbb{R}^n$ at time t, whose state dynamics is governed by a differential equation

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mu), \tag{2.1}$$

where $F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a smooth function, $\mu \in \mathbb{R}$ is a so-called *bifurcation parameter*, and the dot denotes the usual time derivative. For a fixed μ, this defines a particular dynamical system, which, among others, exhibits a particular *fixed point* profile, i.e., a set of points $\{\mathbf{x}^0 \in \mathbb{R}^n \mid F(\mathbf{x}^0) = 0\}$. The existence of fixed points engenders a possibly highly intricate structure of the system state space $\mathbb{R}^n$. Of particular importance are the so-called *stable* and *unstable manifolds*. The stable manifold of a fixed point $\mathbf{x}^0$ is the continuation (forward in time) of the local eigenspaces of the Jacobian $DF|_{\mathbf{x}^0}$ for negative eigenvalues, the unstable manifold is the continuation (backward in time) for positive eigenvalues (see Jetschke 1989, or any good book about dynamical systems for details). The important part of this point is that the structure of the invariant (stable and unstable) manifolds structures the state space in a characteristic way. A very simple example is shown in Fig. 2.1: even in this simple example, the state space is split into four regions. With a more intricate fixed-point (or attractor) structure that profile can be quite more complex.

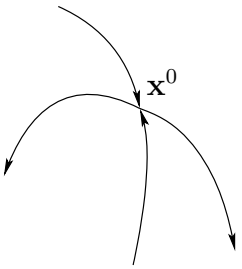


Fig. 2.1. Stable and unstable manifold of a fixed point $\mathbf{x}^0$

The importance of the above observation stems from a number of facts. In the example shown in Fig. 2.1, there are only positive or negative eigenvalues of the Jacobian $DF|_{\mathbf{x}^0}$. However, if we consider μ to be a parameter that is scanned through, the eigenvalue spectrum of the Jacobian changes smoothly, and eigenvalues may change sign, i.e., may cross the 0 level. Generically, an eigenvalue changing sign on changing μ will travel with nonzero speed through 0, i.e., for which $DF_\mu|_{\mathbf{x}^0} \neq 0$, where DF_μ is the partial derivative of the Jacobian with respect to μ. If this is the case, the number or character of the fixed points may change, sometimes dramatically, and with it the whole split of the space into attractor regions of different character. This process is known as *bifurcation*. In systems which have a fast dynamics F parametrized by a slow-varying (and perhaps externally controlled) parameter μ, the appearance of new fixed points is often interpreted as a process of self-organization.

2.4.2 Synergetics

The concept of a slow-varying parameter has been made part of the above analysis by a number of approaches, most notably the synergetics approach (Haken 1983), but it is also known under the names of *slow manifold* and *fast foliation* (Mees 1981). If we consider a dynamical system where the Jacobian of F has a few eigenvalues very close to 0 and a large number of strongly negative eigenvalues, those components of x which fall into the degrees of freedom of the strongly negative eigenvalues will vanish quickly, reducing the essential dynamics of the system to the low-dimensional submanifold of the whole system, which corresponds to its "slow" degrees of freedom. In the more colourful language of synergetics, these "slow" degrees of freedom are the "master" modes that "enslave" the fast, quickly decaying modes that belong to the strongly negative eigenvalues.

Synergetics provided an historically early formal and quantitative approach for the treatment of self-organization phenomena by decomposing a possibly large system into hierarchically separate layers of dynamics. It has been successfully applied to a number of physical systems and models, including laser pumping, superconductivity, the Ginzburg-Landau equations, and the pattern formation and the Bénard instabilities in fluids (Haken 1983). However, it only works properly under certain conditions (namely the particular structure of the eigenvalue spectrum) and there are self-organization phenomena it fails to capture fully (Spitzner and Polani 1998).

2.4.3 Pattern Formation in Spatial Media

To define the dynamical system concept from Sections 2.4.1 and 2.4.2 one requires the concept of smooth manifolds, i.e., a space with consistent differentiable structures. If one adds the requirement that F obey given symmetries, i.e., that there is a symmetry group Γ such that $\gamma \in \Gamma$ operates on $\mathbb{R}^n$ and (2.1) obeys

$$\dot{(\gamma\mathbf{x})} = F(\gamma\mathbf{x}, \mu)$$

for all $\gamma \in \Gamma$, then it can be shown that this imposes restrictions on the solution space, including the bifurcation structure (Golubitsky and Stewart 2003).

A special, but important case is a spatial symmetry governing the state space of the dynamics. In the most general case, the state space is not the finite-dimensional space $\mathbb{R}^n$, but rather the space $C^m(\mathbb{R}^k, \mathbb{R})$ of sufficiently smooth functions on $\mathbb{R}^k$, the symmetries are the Euclidean symmetries (translations and orthogonal rotations) on $\mathbb{R}^k$, and (2.1) actually becomes a partial differential equation.[3] In a discrete approximation one can replace the space $\mathbb{R}^k$ by a *lattice* $L = \{\sum_{i=1}^{k} l_i \mathbf{v}_i \mid l_i \in \mathbb{Z}\}$ for linear independent $\mathbf{v}_i \in \mathbb{R}$ (Hoyle 2006), and thus reobtain a finite-dimensional version of (2.1); this time the n components of space no longer form an unstructured collection, but rather are organized as a lattice and subject to its symmetries. Here, the system dynamics together with the symmetries governing the system give rise to particular stable states and attractors, where, due to their easily visualizable spatial structure, it is easy to detect phenomena of self-organization.

A classical example for such a model is Turing's reaction-diffusion model (Turing 1952). He was among the first to study the dynamics of reaction-diffusion systems as possible models for computation, and his particular interest was pattern formation in biological scenarios. At a time when there was still a debate as to what would be the most adequate model for computation, Turing's suggestion of a spatially organized computational medium with an activator and an inhibitor substance with differing diffusion coefficients provided a wide range of spatial organization dynamics. There are different instances of reaction-diffusion machines, depending on the chemical dynamics. Figure 2.2 shows some Turing patterns emerging from having an activator and an inhibitor with concentrations a, b, respectively, computed from the definition of their rates of change,

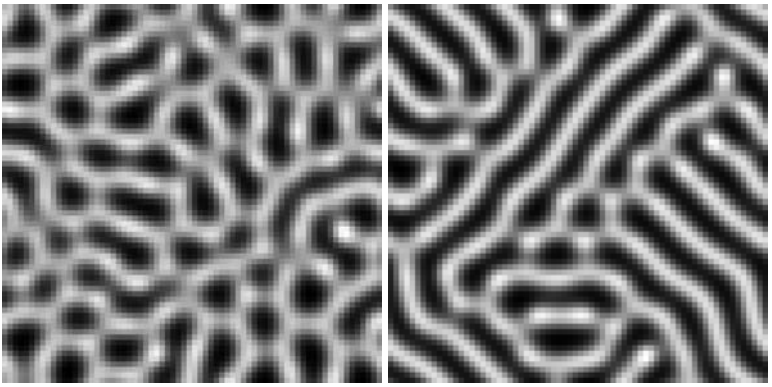


Fig. 2.2. Turing patterns of the concentration a of the activator, as emerging from Eq. (2.2) for different parameters k. See (Bar-Yam 1997) for a discussion on how to obtain different patterns.

[3] Here we ignore technical details necessary to properly define the dynamics.

$$\frac{\partial a}{\partial t} = \delta_1 \Delta a + k_1 a^2/b - k_2 a \tag{2.2}$$

$$\frac{\partial b}{\partial t} = \delta_2 \Delta b + k_3 a^2 - k_4 b, \tag{2.3}$$

in the discrete approximation of a square lattice (Bar-Yam 1997). The starting configurations were randomly initialized concentration fields for a and b. The simulations show that the dynamics is constrained to produce structured patterns from virtually arbitrary initial states.

Other reaction-diffusion systems exhibiting spatial self-organization phenomena are, for instance, the Belousov-Zhabotinsky reaction, which can be implemented in vitro. Reaction-diffusion processes are believed to influence the morphogenetic processes in living things (Meinhardt 1982) and, as such, are of central importance for an understanding how of morphological complexity can be obtained by "unpacking" the relatively compact genotype.

2.5 Information-Theoretic Approaches to Self-organization

The models from Section 2.4 have in common that they require the dynamical system to "live" on a differentiable manifold to describe self-organization. In addition, the synergetics model of self-organization requires a particular grouping of the eigenvalues of the Jacobian, and the pattern formation models require the presence of a spatially structured state space. These are relatively specific requirements. In a unified treatment of self-organization, it would be very limiting to exclude self-organization in discrete, not differentiable, worlds. Similarly, it would be inappropriate to assume that systems must have a Euclidian spatial organization similar to reaction-diffusion systems. It is easy to envisage scenarios where a system may possess other topological/structural properties, such as social or food web networks.

In addition, the example systems from Section 2.4 were implicitly assumed to be deterministic, which is usually far too strong an assumption. Neither this assumption nor the assumption from the last paragraph needs to hold. One can imagine self-organization in an agent system (such as relevant for engineering problems) which is neither deterministic nor organized on a regular spatial structure, and certainly nothing can be assumed in terms of distributions of eigenvalues close to fixed points (if these exist at all).

Can anything at all be analyzed in such structurally impoverished scenarios? Indeed, it turns out that a lot can still be said with much less structure, and the toolbox of choice is information theory. In the following, we outline two approaches for modelling self-organization using information theory.

Information theory operates on probability distributions. These require only minimal structure (a probability measure) on the space of interest and make no assumptions about differentiability or spatial structure. Information theory has crystallized as a promising common language for the study of general systems—to tackle issues of complex phenomena exhibiting a wide variety of seemingly incompatible properties.

2.5.1 Notation

Due to space limitations, the formalization of the exposition is restricted to a minimum. Consider a random variable X assuming values $x \in \mathcal{X}$, $\mathcal{X}$ being the set of possible values for X. For simplicity, assume that $\mathcal{X}$ is finite. Define the *entropy* of X by

$$H(X) := -\sum_{x \in \mathcal{X}} p(x) \log p(x),$$

and the *conditional entropy* of Y as

$$H(Y|X) := \sum_{x \in \mathcal{X}} p(x) H(Y|X = x),$$

with

$$H(Y|X = x) := -\sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x)$$

for $x \in \mathcal{X}$. The *joint entropy* of X and Y is the entropy $H(X, Y)$ of the random variable (X, Y). The *mutual information* of random variables X and Y is $I(X; Y) := H(Y) - H(Y|X) = H(X) + H(Y) - H(X, Y)$. A generalization is the *intrinsic information*: for random variables $X_1, \ldots, X_k$, the intrinsic or multi-information is

$$I(X_1; \ldots; X_k) := \left[\sum_{i=1}^{k} H(X_i)\right] - H(X_1, \ldots, X_k).$$

In Tononi et al. (1994) this notion is also known e.g., as *integration*. Similar to the mutual information, it is a measure of the degree of dependence between the different X_i.

2.5.2 Self-organization as Increasing Statistical Complexity

One influential approach to the study of complex systems and the notions of self-organization and emergence is based on the ϵ-machine formalism, which provides a model to describe complex temporal processes (Crutchfield and Young 1989). Using this formalism, Shalizi (2001) develops a quantifiable notion of self-organization. In the following, we briefly describe the ϵ-machine formalism and the ensuing model for self-organization.

Consider a stochastic process (with, say, infinite past and future):

$$\mathbf{X} = \ldots X^{(t-3)}, X^{(t-2)}, X^{(t-1)}, X^{(t)}, X^{(t+1)}, X^{(t+2)}, X^{(t+3)}, \ldots \ .$$

Denote the (ordered) sequence of variables up to $X^{(t)}$ by $\overleftarrow{X}$ (past) and the sequence of variables from $X^{(t+1)}$ upwards by $\overrightarrow{X}$ (future). Consider the equivalence relation that identifies all pasts $\overleftarrow{x}$ for which the probability distribution $P(\overrightarrow{X} | \overleftarrow{x})$ of the possible futures is exactly the same. This equivalence relation partitions the pasts into disjoint

sets, which, for the sake of simplicity, we name $\tilde{x}$. Any past $\overleftarrow{x}$ is a member of exactly one equivalence class $\tilde{x}$.

To construct an ϵ-machine from a given process $\mathbf{X}$, define an automaton such that its states are identified one-to-one by the equivalence classes $\tilde{x}$ arising from the above procedure. When a transition from t to $t+1$ is made, it means replacing a past $\ldots X^{(t-3)}, X^{(t-2)}, X^{(t-1)}, X^{(t)}$ by a past $\ldots X^{(t-2)}, X^{(t-1)}, X^{(t)}, X^{(t+1)}$, and thus it acts as a transition from an equivalence class $\tilde{x}$ to an equivalence class $\tilde{x}'$, corresponding to the new past. Together with labelling the transition by the realization $x^{(t+1)}$ of $X^{(t+1)}$, this defines the automaton.

The ϵ-machine, when it exists, acts as the unique minimal maximally predictive model of the original process (Shalizi and Crutchfield 2002), including highly non-Markovian processes which may contain a significant amount of memory.[4] It allows one to define the concept of *statistical complexity* as the entropy $H(\tilde{X}) = -\sum_{\tilde{x}} p(\tilde{x}) \log p(\tilde{x})$ of the states of the ϵ-machine. This is a measure of the memory required to perform the process $\mathbf{X}$.

Note that the statistical complexity is, in general, different from another important quantity, known, among other names, as *excess entropy* and is given by $\eta(\mathbf{X}) := I(\overleftarrow{X}; \overrightarrow{X})$ (see, e.g., Grassberger 1986). One always has $\eta(\mathbf{X}) \leq H(\tilde{X})$ (Shalizi 2001). The interpretational distinction between statistical complexity and excess entropy is subtle. Of the selection of interpretations available, the author prefers the view inspired by the "information bottleneck" perspective (Tishby et al. 1999; Shalizi and Crutchfield 2002): The excess entropy is the actual information contained in the complete past (for a given time) about the complete future *as it could be reconstructed if the complete past were available as a whole*. As opposed to that, to obtain the statistical complexity one has to force this information through the "bottleneck" given by the ϵ-machine state at the *present* time slice, which has to provide sufficient statistics about the past concerning the future constrained to the current time. Because of the constraint of this bottleneck to the present time slice it is, in general, less parsimonious in description than the "idealized" excess entropy, which, in principle, assumes availability of the whole process (past and future) to produce its prediction. In the bottleneck picture, we have the sequence

$$\tilde{X} \longleftarrow \overleftarrow{X} \longleftrightarrow \overrightarrow{X},$$

where the left arrow indicates the projection of $\overleftarrow{X}$ to the ϵ-machine state and the arrows between the past and future variables indicate their informational relation. The process of "squeezing" their mutual information into the bottleneck variable $\tilde{X}$ produces in general a variable with a larger entropy than the actual mutual information content between $\overleftarrow{X}$ and $\overrightarrow{X}$. (This is a general property of the information bottleneck, of which the relation between statistical complexity and excess entropy is just a special case.)

In computing the ϵ-machine, the picture of an infinite time series is idealized. In the empirical case one would consider finite time series, giving rise to a "localized"

[4] Note that, in general, the construction of an ϵ-machine from the visible process variables $\mathbf{X}$ is not necessarily possible, and the reader should be aware that the Shalizi/Crutchfield model is required to fulfil suitable properties for the reconstruction to work. I am indebted to Nihat Ay and Wolfgang Löhr for pointing this out to me.

ϵ-machine, operating on a certain window size. Here, one would expect to encounter a slow drift superimposed on the fast dynamics of the process, which will slowly change the localized ϵ-machine with the passing of time. Shalizi (2001) calls a system self-organizing if this statistical complexity grows with time. We will call this flavour of self-organization *SC-self-organization* ('SC' for statistical complexity).

This approach basically considers organization to be essentially the same as complexity. In this model, self-organization is an intrinsic property of the system and unambiguously measurable. In particular, this approach makes the relation to emergence unambiguously clear, as Shalizi gives a definition of emergence based on the ϵ-machine notion in the same work. In essence in the ϵ-machine perspective, emergence is present if there is a coarse-grained description of the system that is more predictively efficient than the original description of the process, i.e., if it has a higher ratio $\eta(\mathbf{X})/H(\tilde{X})$ of excess entropy vs. statistical complexity, a better ratio between the total amount of process information that ideally needs to be processed and the process memory that is actually necessary to achieve it.

This approach to emergence is descriptive, as it characterizes a property of the particular description (i.e., perspective or "coordinate system") through which one looks into a system. As opposed to that, self-organization in the ϵ-machine model is a purely intrinsic property of the system. Through this split into description and intrinsic properties, Shalizi (2001) argues that while emergence may allow one to simplify descriptions of a system, there may be cases of self-organization which humans do not recognize as such because there is no appropriate simplified (emergent) coordinate system through which the self-organization would become apparent. It is only visible through the ϵ-machine construction. This provides a transparent picture how self-organization and emergence turn out to be two mathematically distinct concepts that represent different aspects of a system. While this is a motif that one finds repeatedly emphasized in the scientific discourse, it is rarely formulated in such compelling and crisp language.

One interesting aspect of the ϵ-machine view is how it reflects the bifurcation concept from Section 2.4.1. Consider as process an iterator map for $t \to \infty$. As long as there is only a single fixed-point attractor, the ϵ-machine will (asymptotically) have only one state. As a bifurcation into two fixed-point attractors occurs, these two attractors will be reflected by the ϵ-machine. With the bifurcation behaviour becoming more intricate (as would happen, say, in the logistic map example with an adiabatically slowly growing bifurcation parameter), the ϵ-machine also grows in complexity. In this way, the ϵ-machine can grow significantly in size and in statistical complexity.

2.5.3 Observer-Induced Self-organization

Conceptually, the pattern formation which we gave in Section 2.4.3 as a prominent example of self-organization does not fit smoothly into the picture of growing statistical complexity. One reason is that statistical complexity by its very foundation is a concept that operates on a process that has no a priori structure on the $X^{(t)}$, except for the ordered-ness (and, implied, directed-ness) of time.

Quite different from that, spatial pattern formation inextricably requires a spatial structure on its state space $\mathcal{X}$. The patterns that develop during the experiments form in space, and space has strong structural constraints, So Shalizi (2001), spatial ϵ-machine developed a to deal specifically with this problem. Thus, the spatial structure is brought in explicitly as part of the model.

An alternative approach to modelling self-organization using information theory is suggested in Polani (2003). This approach no longer considers an unstructured dynamical system on its own, but adds the concept of an *observer* which acts as a particular "coordinate system" through which the given system is represented at a given time step. For this model of self-organization, an observer or coordinate system needs to be specified *in addition* to the dynamics of the system. The suspicion that observers may be of importance in characterizing complex systems has been voiced quite a few times in the past (Crutchfield 1994; Baas and Emmeche 1997; Harvey 2000; Rasmussen et al. 2001). In formalizing this idea here, we follow the particular flavour from Polani (2003).

2.6 Organization via Observers

A *(perfect) observer* of a (random) variable X is a collection $X_1, X_2, \ldots, X_k$ of random variables allowing full reconstruction of X, i.e., for which $H(X|X_1, X_2, \ldots, X_k)$ vanishes. We define the *organization information* with respect to the observer as the multi-information $I(X_1; \ldots; X_k)$. We call a system *self-organizing* (with respect to the given observer) if the organization information increase with respect to the observer variables is positive as the system dynamics progresses with time. $I(X_1; \ldots; X_k)$ quantifies to what extent the observer variables $X_1, X_2, \ldots, X_k$ depend on each other. We call this flavour of self-organization *O-self-organization* (O for observer based).

The set of observer variables can often be specified in a natural way. For instance, systems that are composed of many, often identical, individual subsystems have a canonical observer, defined via the partition of the system into subsystems. For instance, the observer variables could denote the states of agents that collectively make up a system. An increase in the multi-information of the system with respect to the agent states then indicates an increasing degree of coordination among the agents, this is consistent with our intuitive understanding of self-organization. Reaction-diffusion systems are also naturally described in this framework. Each point in space becomes an observer variable; in the attractor states with spot-and-wave patterns, these observer variables are intrinsically correlated.

Note, however, that for the multi-information not to vanish, it is still necessary that the whole system have some degree of freedom and that there is not just a single fixed pattern that the system converges to. This makes sense, since otherwise one would just be talking about a single attractor, and thus trivial, system.

Using the self-organizing map as the model system and the individual neuron weights as observer variables, Polani (2003) discusses in detail the advantage of multi-information as a measure for self-organization, as compared to other information-

theoretic candidates for such a measure (except for the comparison with SC-self-organization, which is discussed in the present chapter for the first time). Many of the arguments carry over to other typical scenarios.

Note that compared to SC-self-organization (Sec. 2.5.2), O-self-organization is different in several respects. SC-self-organization does not require observers, and arises from the intrinsic dynamics of the system. This is orthogonal to the view of O-self-organization. In principle, the need for fewer assumptions by SC-self-organization is a conceptual advantage. On the other hand, to model the appearance of regular patterns (e.g., of a reaction-diffusion system) as a self-organization process one must in any case specify the extra spatial structure in which the patterns appear. In O-self-organization, this can be directly made part of the specification. Thus, O-self-organization would be a natural candidate for these types of scenarios.

2.6.1 Observer Dependence

For the observer-based measure, a central question is how the measure changes as one moves from one observer to another; i.e., what happens to the measure on a change of the "coordinate system." It turns out that it is possible to formulate an interesting relation between fine-grained observers and a coarse-graining of the same observers. We discuss this relation in the following.

Let $X_i, i = 1 \ldots k$ be a collection of jointly distributed random variables; this collection forms the *fine-grained observer*. Obtain the *coarse-grained observer* by grouping the X_i according to

$$\underbrace{X_1, \ldots, X_{k_1}}_{\tilde{X}_1}, \underbrace{X_{k_1+1}, \ldots, X_{k_2}}_{\tilde{X}_2}, \ldots, \underbrace{X_{k_{\tilde{k}-1}+1}, \ldots, X_k}_{\tilde{X}_{\tilde{k}}}. \tag{2.4}$$

That is, each of the coarse-grained variables $\tilde{X}_j, j = 1 \ldots \tilde{k}$ is obtained by grouping several of the fine-grained variables X_i together.

Then the multi-information of the fine-grained observer can be expressed as[5]

$$I(X_1; X_2; \ldots; X_k) = I(\tilde{X}_1; \tilde{X}_2; \ldots; \tilde{X}_{\tilde{k}}) + \sum_{j=1}^{\tilde{k}} I(X_{k_{j-1}+1}; \ldots; X_{k_j}), \tag{2.5}$$

(where we adopt the convention $k_0 := 0$ and $k_{\tilde{k}} := k$). Relation (2.5) states that the multi-information as a measure of self-organization in the fine-grained case can be expressed as the multi-information for the set of coarse-grained variables, corrected by the *intrinsic* multi-information of all these coarse-grained variables, or in other words, "the fine-grained system is more than the sum of its coarse-grained version." The proof is sketched in the appendix.[6]

[5] This is a generalization of Eq. (3) from Tononi et al. (1994) for the bipartite case to the multipartite case.

[6] This property is related to a property that can be proven for graphical models, see, e.g., Proposition 2.1 in Slonim et al. (2001).

Equation (2.5) expresses the way that the intrinsic information of a system changes under a "change of coordinates" by regrouping the random variables that represent the system. This "bookkeeping" of multi-information, while changing the basis for the system description in general, only applies to regrouping of variables, but not to recoding. Under recoding of variables (i.e., re-representing the variables X_i by random variables $Y_i = f_i(X_1, \ldots, X_i, \ldots, X_k)$, where f_i is some deterministic function), there is no canonical way of transforming multi-information in a simple and transparent manner.

To see that, note that recoding may entirely remove dependencies between the recoded X_i (e.g., in independent component analysis; Comon 1991). In fact, further requirements can be added to the component independence; indeed, this has been proposed as a way of discovering degrees of freedom representing *emergent levels of description* (Polani 2004, 2006). In this sense, O-self-organization is distinct from and effectively conjugate to the "emergent descriptions" concept. This dichotomy mirrors the analogous dichotomy exhibited by the formalization of self-organization and emergence using the ϵ-machine formalism (Sec. 2.5.2).

2.7 Discussion

2.7.1 SC- and O-Self-organization

We have emphasized that self-organization is a phenomenon that is often discussed in conjunction with complex systems. While there is a manifold selection of processes that are associated with this phenomenon, most notions used to characterize self-organization are either too vague to be useful or too specific to be transferable from one system to another. The information-theoretic notions of (statistical complexity) SC-self-organization (Shalizi 2001; Shalizi et al. 2004) and that of (observer-based) O-self-organization (Polani 2003) strive to fill this gap. Although similar to each other in the general philosophy, they are essentially "orthogonal" as quantities.

SC-self-organization takes in a single time series and measures the growth in statistical complexity with time. O-self-organization requires an observer, i.e., a set of variables through which the system state is observed. Such a set of variables is often naturally available, for instance, in multiagent systems. Similar to SC-self-organization, O-self-organization seems to capture essential aspects of self-organization—for instance, the freezing of seemingly unrelated degrees of freedom (the observer variables) into highly coordinated global behavior.

Whereas SC-self-organization concentrates on the complexity of the temporal dynamics, O-self-organization concentrates on the compositional aspects of the system (this compositionality can, but need not, be spatial). This distinction is also what indicates the use of each of the measures. If one focuses on the temporal dynamics, SC-self-organization may be more relevant; whereas for spatial or compositional dynamics, O-self-organization may be the measure of choice. As the precise mathematical conceptualizations of self-organization are relatively recent, it will take some time until enough experience is accumulated to make an informed decision as to which

measure is appropriate to use in a given constellation, or whether still other, more suitable measures need to be discovered.

A word of caution at this point: the calculation of multi-information is difficult if the number of variables is large (Slonim et al. 2005). Particularly in unorganized and random states, a Monte-Carlo estimate of the entropies and multi-information is likely to be undersampled and to overestimate the organization of the system in that state. Research is under way to develop general methods that are able to properly estimate the organization of unstructured states (and to distinguish them from organized states).

2.7.2 Introducing Observers

For O-self-organization, we have assumed the existence of natural observers. What if none exist? Which ones should be introduced and used? The multi-information term consists of the entropies of the individual variables as well as the entropy of the joint variables. The latter depends only on the total system, not the observer. It is therefore the entropies of the *individual* variables that will change if we choose different observers. In general, it is quite possible to choose observers in such a way as to make them independent, but while this choice of observer is interesting (it essentially corresponds to independent component analysis, Sec. 2.6.1), it makes the system maximally un-self-organized. This clearly shows that O-self-organization is not intrinsic. It is rather "in the eye of the beholder" (Harvey 2000), but in a formally precise way.

Now, for O-self-organization to be present at all, the whole system must have some degree of uncertainty; otherwise the individual variable entropies will collapse and the multi-information will vanish. This is a property of the whole system. Thus, one could consider a natural observer to be one that *maximizes* the multi-information (as opposed to minimizing it), thus making the system as self-organized as possible. If this is the case, O-self-organization could be viewed as the opposite of independent component decomposition.

But there is yet another way of constructing a natural observer: if one considers units (agents) that operate in the system and possess sensors and actuators, the former attaining information about the system and the latter acting upon and modifying the system, then the perception-action loop of these agents forms a structured information channel. It can be shown (Klyubin et al. 2004) that maximizing the information flow through this channel allows the units to extract features from the system that are pertinent to its structure.

This view is particularly interesting since it does not look at a system with a preordained temporal dynamics, but rather the units (agents) have the option of choosing their own actions. Nevertheless, once they perform the information flow maximization, they attain perceptual properties especially appropriate for the system at hand. The thus attained filters or feature detectors could act as another form of natural observer variables for the given system. Similarly, principles of informational organization can lead to joint coordination of a sensorimotor device (Klyubin et al. 2005; Prokopenko et al. 2006) and direct systems to equipment with embodiment-appropriate pattern detector loops.

2.8 Conclusion

The present chapter discussed the general problem of defining self-organization and presented two operational approaches, both based on information-theoretic principles. One approach, based on the ϵ-machine formalism, defines self-organization as an intrinsic property of a system—as a growth of the memory required to process a time series of random variables. The other approach defines self-organization via an observer, in typical cases realized as a family of variables of more-or-less similar type; a growing coordination among these variables with time is then identified as self-organization. Depending on one's aims, one will choose one or the other model to identify self-organization in a system. In particular, SC-self-organization will be the approach of choice if one is interested in characterizing the increase in complexity of the temporal dynamics, whereas O-self-organization emphasizes the self-organization process in a system composed of many individual subsystems.

The advantage of using information-theoretic notions for quantifying self-organization is that they provide a precise language for identifying the conditions of self-organization and the underlying assumptions, as opposed to vague or uncomputable qualitative characterizations. The quantitative character of information measures also allows one to actively search for "more self-organized" systems in a given context, rather than to just state whether a system possesses or does not possess this property (as, e.g., an algebraic characterization would do). In addition, the information-theoretic language forces one to specify the assumptions and requirements underlying the approaches being used.

In short, information theory proves to be a powerful language for expressing self-organization and other concepts relevant to complex systems. Even if one ultimately should prefer a different route to the characterization of self-organization in a complex system, it is probably a good first bet to strive towards a formulation that profits from the clarity and transparence of the information-theoretic language.

2.9 Appendix: Proof of the Relation between Fine- and Coarse-Grained Multi-information

Proof. First, note that a different way to write the composite random variables $\tilde{X}_j$ is $\tilde{X}_j = (X_{k_{j-1}+1}, \ldots, X_{k_j})$ for $j = 1 \ldots \tilde{k}$, giving

$$H(\tilde{X}_j) = H(X_{k_{j-1}+1}, \ldots, X_{k_j}). \tag{2.6}$$

Similarly, the joint random variable $(\tilde{X}_1, \ldots, \tilde{X}_{\tilde{k}})$ consisting of the composite random variables $\tilde{X}_j$ can be seen as a regrouping of the elementary random variables $X_1, \ldots, X_k$. Therefore the joint random variable constructed from the $\tilde{X}_j$ and that constructed from the X_i both have the same entropy:

$$H(\tilde{X}_1, \ldots, \tilde{X}_{\tilde{k}}) = H(X_1, \ldots, X_k). \tag{2.7}$$

For consistency of notation, one writes $k_0 = 0$ and $k_{\tilde{k}} = k$. One then obtains

$$
\begin{aligned}
&I(\tilde{X}_1; \tilde{X}_2; \ldots; \tilde{X}_{\tilde{k}}) + \sum_{j=1}^{\tilde{k}} I(X_{k_{j-1}+1}; \ldots; X_{k_j}) \\
&\quad = \sum_{j=1}^{\tilde{k}} H(\tilde{X}_j) - H(\tilde{X}_1, \ldots, \tilde{X}_{\tilde{k}}) + \sum_{j=1}^{\tilde{k}} \Big(\sum_{j'=k_{j-1}+1}^{k_j} H(X_{j'}) - H(X_{k_{j-1}+1}, \ldots, X_{k_j}) \Big) \\
&\quad = \underbrace{\sum_{j=1}^{\tilde{k}} \sum_{j'=k_{j-1}+1}^{k_j} H(X_{j'})}_{=\sum_{i=1}^{k} H(X_i)} + \sum_{j=1}^{\tilde{k}} \underbrace{\Big(H(\tilde{X}_j) - H(X_{k_{j-1}+1}, \ldots, X_{k_j}) \Big)}_{=0} - \underbrace{H(\tilde{X}_1, \ldots, \tilde{X}_{\tilde{k}})}_{=H(X_1, \ldots, X_k)},
\end{aligned}
$$

where the first term results from a regrouping of summands, the second term results from Eq. (2.6), and the third from rewriting the whole set of random variables from the coarse-grained to the fine-grained notation, thus giving

$$
\begin{aligned}
&= \sum_{i=1}^{k} H(X_i) - H(X_1, \ldots, X_k) \\
&= I(X_1; \ldots; X_k),
\end{aligned}
$$

which proves the equation.

References

Adami, C. (1998). *Introduction to Artificial Life*. Springer, New York.

Ashby, W. R. (1947). Principles of the self-organizing dynamic system. *J. Gen. Psychol.*, 37:125–128.

Ay, N., and Krakauer, D. C. (2006). Information geometric theories for robust biological networks. *J. Theor. Biology*. 125(2):93–121.

Ay, N., and Polani, D. (2007). Information flows in causal networks. *Advances in Complex Systems*. In Press.

Ay, N., and Wennekers, T. (2003). Dynamical properties of strongly interacting Markov chains. *Neural Networks*, 16(10):1483–1497.

Baas, N. A., and Emmeche, C. (1997). On emergence and explanation. *Intellectica*, 2(25): 67–83.

Bar-Yam, Y. (1997). *Dynamics of Complex Systems*. Studies in Nonlinearity. Westview, Boulder, CO.

Bennett, C. H., and Landauer, R. (1985). The fundamental limits of computation. *Scientific American*, pages 253(1):48–56.

Bertschinger, N., Olbrich, E., Ay, N., and Jost, J. (2006). Autonomy: An information theoretic perspective. In *Proceedings of the Workshop on Artificial Autonomy at Alife X, Bloomington, IN*, pages 7–12.

Comon, P. (1991). Independent component analysis. In *Proceedings of the International Signal Processing Workshop on Higher-order Statistics, Chamrousse, France*, pages 111–120.

Crutchfield, J. P. (1994). The calculi of emergence: Computation, dynamics, and induction. *Physica D*, pages 11–54.

Crutchfield, J. P., and Young, K. (1989). Inferring statistical complexity. *Phys. Rev. Lett.*, 63:105–108.

Emmeche, C., Køppe, S., and Stjernfelt, F. (2000). Levels, emergence, and three versions of downward causation. In Andersen, P. B., Emmeche, C., Finnemann, N. O., and Christiansen, P. V., editors, *Downward Causation. Minds, Bodies and Matter*, pages 13–34. Aarhus University Press, Århus.

Golubitsky, M., and Stewart, I. (2003). *The Symmetry Perspective*. Birkhäuser, Basel.

Grassberger, P. (1986). Toward a quantitative theory of self-generated complexity. *Int. J. Theor. Phys.*, 25:907–938.

Haken, H. (1983). *Advanced synergetics*. Springer, Berlin.

Harvey, I. (2000). The 3 Es of artificial life: Emergence, embodiment and evolution. Invited talk at Artificial Life VII, 1-6. August, Portland, Oregon, USA.

Helbing, D., Buzna, L., Johansson, A., and Werner, T. (2005). Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation Science*, 39(1): 1–24.

Hoyle, R. (2006). *Pattern Formation*. Cambridge University Press, Cambridge.

Jetschke, G. (1989). *Mathematik der Selbstorganisation*. Vieweg, Braunschweig.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2004). Organization of the information flow in the perception-action loop of evolved agents. In *Proceedings of 2004 NASA/DoD Conference on Evolvable Hardware*, pages 177–180. IEEE Computer Society.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). Empowerment: A universal agent-centric measure of control. In *Proceedings of the IEEE Congress on Evolutionary Computation, 2–5 September 2005, Edinburgh, Scotland (CEC 2005)*, pages 128–135. IEEE.

Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5:183–191.

Mees, A. I. (1981). *Dynamics of feedback systems*. Wiley, New York.

Meinhardt, H. (1972). A theory of biological pattern formation. *Kybernetik*, 12:30–39.

Meinhardt, H. (1982). *Models of Biological Pattern Formation*. Academic, New York.

Pask, G. (1960). The natural history of networks. In Yovits, M. C., and Cameron, S., editors (1960). *Self-Organizing Systems – Proceedings of an Interdisciplinary Conference, on Computer Science and Technology and their Application, 5–6 May 1959*. Pergamon, New York.

Polani, D. (2003). Measuring self-organization via observers. In Banzhaf, W., Christaller, T., Ziegler, J., Dittrich, P., Kim, J. T., Lange, H., Martinetz, T., and Schweitzer, F., editors, *Advances in Artificial Life (Proceedings of the 7th European Conference on Artificial Life, Dortmund, September 14–17, 2003)*, Springer, Berlin, Heidelberg.

Polani, D. (2004). Defining emergent descriptions by information preservation. *InterJournal Complex Systems*, 1102.

Polani, D. (2006). Emergence, intrinsic structure of information, and agenthood. *InterJournal Complex Systems*, 1937.

Prigogine, I., and Nicolis, G. (1977). *Self-Organization in Non-Equilibrium Systems: From Dissipative Structures to Order Through Fluctuations*. Wiley, New York.

Prokopenko, M., Gerasimov, V., and Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J. C. T., Marocco, D., Meyer, J.-A., Miglino, O., and Parisi, D., editors, *From Animals to Animats 9: 9th International Conference on the Simulation of Adaptive Behavior (SAB 2006), Rome, Italy*, volume 4095 of *Lecture Notes in Computer Science*, pages 558–569. Springer. Berlin, Heidelberg.

Rasmussen, S., Baas, N., Mayer, B., Nilsson, M., and Olesen, M. W. (2001). Ansatz for dynamical hierarchies. *Artificial Life*, 7:329–353.

Reichl, L. (1980). *A Modern Course in Statistical Physics*. University of Texas Press, Austin.

Shalizi, C. R. (2001). *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Wisconsin, Madison.

Shalizi, C. R., and Crutchfield, J. P. (2002). Information bottlenecks, causal states, and statistical relevance bases: How to represent relevant information in memoryless transduction. *Advances in Complex Systems*, 5(1):91–95.

Shalizi, C. R., Shalizi, K. L., and Haslinger, R. (2004). Quantifying self-organization with optimal predictors. *Physical Review Letters*, 93(11):118701.

Slonim, N., Friedman, N., , and Tishby, T. (2001). Agglomerative multivariate information bottleneck. In *Neural Information Processing Systems (NIPS 01)*, pages 929–936, La Jolla.

Slonim, N., S. Atwal, G., Tkačik, G., and Bialek, W. (2005). Estimating mutual information and multi-information in large networks. arXiv:cs.IT/0502017.

Spitzner, A., and Polani, D. (1998). Order parameters for self-organizing maps. In Niklasson, L., Bodén, M., and Ziemke, T., editors, *Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN 98), Skövde, Sweden*, volume 2, pages 517–522. Springer, Berlin.

Tishby, N., Pereira, F. C., and Bialek, W. (1999). The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing, IL*, Urbana-Champaign.

Tononi, G., Sporns, O., and Edelman, G. M. (1994). A measure for brain complexity: Relating functional segregation and integration in the nervous system. *Proceedings of the Natural Academy of Sciences, U.S.A.*, 91:5033–5037.

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. B*, 327:37–72.

Walter, W. G. (1951). A machine that learns. *Scientific American*, pages 185(2):60–63.

Yovits, M. C., and Cameron, S., editors (1960). *Self-Organizing Systems – Proceedings of an Interdisciplinary Conference, on Computer Science and Technology and their Application, 5-6 May 1959*. Pergamon, New York .

Part II

Distributed Management and Control

3

Self-Organizing Traffic Lights: A Realistic Simulation

Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe

3.1 Introduction: Catch the Green Wave? Better Make Your Own!

Everybody in populated areas suffers from traffic congestion problems. To deal with them, various methods have been developed to mediate between road users as well as possible. Traffic lights are not the only pieces in this puzzle, but they are an important one. As such, different approaches have been used in attempts to reduce user waiting times and prevent traffic jams. The most common involves finding the appropriate phases and periods of traffic lights to quantitatively optimize traffic flow. This results in "green waves" that flow through the main avenues of a city, ideally enabling cars to drive through them without facing a red light, as the speed of the green wave matches the desired cruising speed for the avenue. However, this approach does not consider the current state of the traffic. If there is high traffic density, cars entering a green wave will be stopped by cars ahead of them or cars that turned into the avenue, and once a car misses the green wave, it has to wait the whole duration of the red light to get into the next green wave. On the other hand, for very low densities, cars might arrive at the next intersection too quickly, and then to stop at each crossing. This method is certainly better than having no synchronization at all; however, it can be greatly improved.

Traffic modeling has greatly enhanced our understanding of this complex phenomenon, especially during the last decade (Prigogine and Herman 1971; Wolf et al. 1996; Schreckenberg and Wolf 1998; Helbing 1997; Helbing and Huberman 1998; Helbing et al. 2000), suggesting various improvements in traffic infrastructure. One of these consists of adapting the traffic lights to the current traffic conditions. Indeed, modern "intelligent" advanced traffic management systems (ATMS) use learning methods to adapt phases of traffic lights, normally employing a central computer (Federal Highway Administration 1998; Hunt et al. 1981). The self-organizing approach we present here does not need a central computer, as the global synchronization is adaptively achieved by local interactions between cars and traffic lights, generating flexible green waves on demand.

We have previously shown in an abstract simulation (Gershenson 2005) that self-organizing traffic lights can greatly improve traffic flow for any density. In this chapter, we extend these results to a realistic setting, implementing self-organizing traffic lights

in an advanced traffic simulator using real data from a Brussels avenue. In the next section, we give a brief introduction to the concept of self-organization. The SOTL control method is then presented, followed by the moreVTS simulator. In Section 3.5, results from our simulations are shown, followed by discussion, future work, and conclusions.

3.2 Self-Organization

The term *self-organization* has been used in different areas with different meanings, as is cybernetics (von Foerster 1960; Ashby 1962), thermodynamics (Nicolis and Prigogine 1977), biology (Camazine et al. 2003), mathematics (Lendaris 1964), computing (Heylighen and Gershenson 2003), information theory (Shalizi 2001), synergetics (Haken 1981), and others (Skår and Coveney 2003). (For a general overview, see [(Heylighen 2003).] However, the use of the term is subtle, since any dynamical system can be said to be self-organizing or not, depending partly on the observer (Ashby 1962; Gershenson and Heylighen 2003): If we decide to call a "preferred" state or set of states (i.e., attractor) of a system "organized," then the dynamics will lead to a self-organization of the system.

It is not necessary to enter into a philosophical debate on the theoretical aspects of self-organization to work with it, so a practical notion will suffice (Gershenson 2006):

> A system *described* as self-organizing is one in which elements *interact* in order to achieve *dynamically* a global function or behavior.

This function or behavior is not imposed by one single or several elements, nor is it determined hierarchically. It is achieved *autonomously* as the elements interact with one another. These interactions produce feedbacks that regulate the system. If we want the system to solve a problem, it is useful to describe a complex system as self-organizing when the "solution" is not known beforehand and/or is changing constantly. Then, the solution is dynamically sought by the elements of the system. In this way, systems can adapt quickly to unforeseen changes as elements interact locally. In theory, a centralized approach could also solve the problem, but in practice such an approach would require too much time to compute the solution and would not be able to keep pace with the changes in the system and its environment.

In engineering, a self-organizing system would be one in which elements are designed to *dynamically* and *autonomously* solve a problem or perform a function at the system level. Our traffic lights are self-organizing because each one makes a decision based only on local information concerning its own state. Still, they manage to achieve robust and adaptive global coordination.

3.3 Self-Organizing Traffic Lights: The Control Method

In the SOTL method [originally named *SOTL-platoon* in Gershenson (2005)], each traffic light, i.e., intersection, keeps a counter κ_i that is set to zero when the light turns red and then incremented at each time step by the number of cars approaching *only*

the red light (i.e., the next one a car will reach) independently of the status or speed of the cars (i.e., moving or stopped). When κ_i (representing the integral of cars over time) reaches a threshold θ, the green light at the same intersection turns yellow, and at the following time step it turns red with $\kappa_i = 0$, while the red light that counted turns green. In this way, if there are more cars approaching or waiting behind a red light, it will turn to green faster than if there are only few cars. This simple mechanism achieves self-organization in the following way: if there are only a few cars, these will be stopped behind red lights for more time, giving other cars time to join them. As more cars join the group, cars will wait less time behind red lights. With a sufficient number of cars, the red lights will turn green even before they reach the intersection, generating "green corridors." Having "platoons" or "convoys" of cars moving together improves traffic flow, compared to a homogeneous distribution of cars, since there are large empty areas between platoons, which can be used by crossing platoons with little interference.

The following constraint prevents traffic lights from switching too fast when there are high densities: a traffic light will not change if the number of time steps is less than a minimum phase, i.e., $\varphi_i < \varphi_{\min}$ (φ_i is the time since the light turned green).

Two further conditions are taken into account to regulate the size of the platoons. Before changing a red light to green, the controller checks if a platoon is crossing through, in order not to break it. More precisely, a red light is not changed to green if on the crossing street there is at least one car approaching within a distance ω from the intersection. This keeps crossing platoons together. For high densities, this condition alone would cause havoc, since large platoons would block the traffic flow of intersecting streets. To avoid this, we introduce a second condition: condition one is not taken into account if there are more than μ cars approaching the green light. Thus, long platoons can be broken, and the restriction only comes into play if a platoon will soon be through an intersection.

The SOTL method is formally summarized in Algorithm 3.1.

This method has no phase or internal clock. If there are no cars approaching a red light, the complementary light can stay green. We say that this method is self-organizing because the global performance is given by the local rules followed

Algorithm 3.1: Self-organizing traffic lights (SOTL) controller.

```
1:  foreach (time step) do
2:      κ_i += cars_approachingRed in ρ
3:      if (φ_i ≥ φ_min) then
4:          if not (0 < cars_approachingGreen in ω < μ) then
5:              if (κ_i ≥ θ) then
6:                  switchlight_i()
7:                  κ_i = 0
8:              end
9:          end
10:     end
11: end
```

by each traffic light: they are "unaware" of the state of other intersections and still manage to achieve global coordination.

The method employs a similar idea to the one used by Porche and Lafortune (1998), but with a much simpler implementation. There is no costly prediction of arrivals at intersections, no need to establish communication between traffic lights to achieve coordination, and not fixed cycles.

3.4 A Realistic Traffic Simulator: moreVTS

Our simulator (moreVTS 2006), (a more realistic Vehicle Traffic Simulator) is the third in a series of open source projects building on the previous one, developed in Java. Green Light District (GLD 2001) was developed by the Intelligent Systems Group at the University of Utrecht (Wiering et al. 2004). It was then improved upon by students in Argentina within the iAtracos project, which we used as a starting point for our simulator, which introduces realistic physics into the simulation. Among other things, acceleration was introduced and the scale was modified so that one pixel represents 1 m and one cycle represents 1 s.

The simulator allows the modeling of complex traffic configurations, enabling the user to create maps and then run simulations varying the densities and types of road users. Multiple-lane streets and intersections can be arranged, as well as spawn and destination frequencies of cars. For implementation details of moreVTS, the reader is referred to Cools (2006).

The self-organizing traffic light controller described in the previous section was implemented in moreVTS. Using data provided by the Brussels Capital Region, we were able to build a detailed simulation of the Rue de la Loi/Wetstraat, a four-lane one-way westward avenue in Brussels that gathers heavy traffic toward the center of the city. We used the measured average traffic densities per hour on working days for 2004 (shown in Table 3.1) and the current "green wave" method, which has a period of 90 s, with 65 s for the green phase on the Wetstraat, 19 for the green phase on side streets, and 6 for transitions. This enabled us to compare our self-organizing controller with a standard one in a realistic setting. Figure 3.1 shows the simulation view of the Wetstraat and its surrounding streets.

The data from Table 3.1 is for the cars entering the Wetstraat on the east, so the spawn rates for the two nodes in the simulation representing this were set according to these data. For the other nodes, the spawn and destination frequencies were set based on a field study we performed in May 2006, comparing the percentage of cars

0	1	2	3	4	5	6	7	8	9	10	11
476	255	145	120	175	598	2933	5270	4141	4028	3543	3353

12	13	14	15	16	17	18	19	20	21	22	23
3118	3829	3828	3334	3318	3519	3581	3734	2387	1690	1419	1083

Table 3.1. Average vehicle count per hour at the beginning of the Wetstraat. Data kindly provided by the Brussels Capital Region.

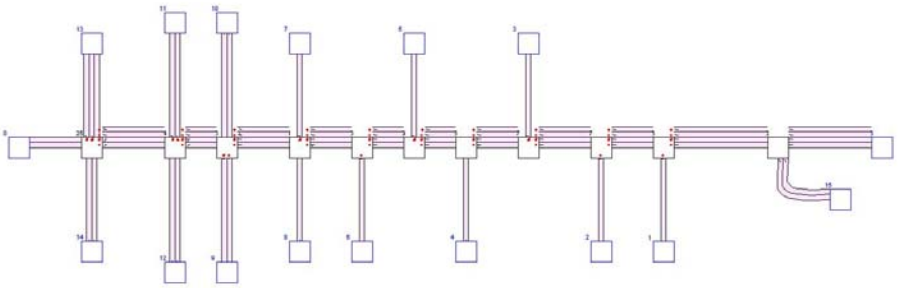


Fig. 3.1. Simulation of the Wetstraat and intersecting streets. Cars flow westward on the Wetstraat. Dots represent traffic lights for each incoming lane at intersections.

that flow through the Wetstraat and those that flow through side streets, entering or leaving the Wetstraat. These percentages were kept constant, so that when the density of cars entering the Wetstraat changed, all the other spawn rates changed in the same proportion. On average, for every five cars flowing through a side street, 100 flow through the Wetstraat. This is not the case of the Kuststraat, a two-way avenue at the west of the Wetstraat (second and third crossing streets from left to right in Fig. 3.1), where for 100 cars driving through the Wetstraat, about 40 turn right, 40 turn left, and only 20 go straight, while 20 more drive through the Kuststraat (about 10 in each direction). The precise spawn rates and destination frequencies are given in Cools (2006, pp. 55–57).

3.5 Results

To measure the performance of the current green wave method and our self-organizing controller, we used the average trip waiting times (ATWT). The trip waiting time for one car is the travel time minus the minimum possible travel time (i.e., travel distance divided by the maximum allowed speed, which for the Wetstraat simulation is about 60 s).

Several simulation runs were performed to find the best parameters for the SOTL method. For each parameter and traffic density, five simulation runs representing 1 h, i.e., 3600 cycles, were averaged. The results were robust and consistent, with SOTL performing better than the green wave method for a wide range of parameters θ and $\varphi_{\min}$ (Cools 2006). Only the best results are shown in Fig. 3.2, together with the results for the green wave method. The cruise speed used was 14 m/s, $\omega = 25$ and $\mu = 3$. Since some densities from Table 3.1 are very similar, we averaged and considered the same densities for 2:00, 3:00, and 4:00; 8:00 and 9:00; 10:00, 17:00, and 18:00; 11:00, 15:00, and 16:00; 13:00, 14:00 and 19:00; and 21:00 and 22:00.

As Fig. 3.2 shows, there is considerable reduction in ATWT using SOTL instead of the current green wave method. The ATWT for the densities at different hours using SOTL were from 34 to 64% of the ATWT for the green wave method, and on average

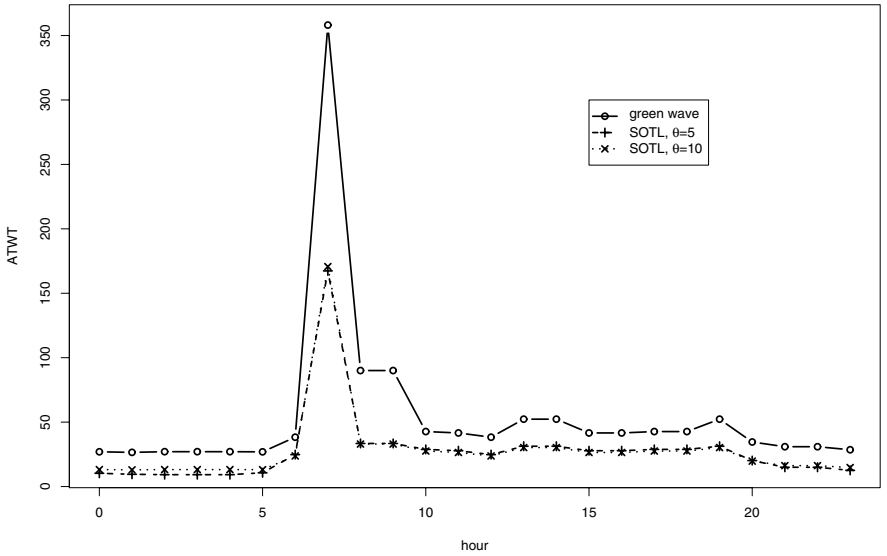


Fig. 3.2. Average trip waiting times (ATWT) at different hours of the day with green wave and SOTL controllers with $\varphi_{\min} = 5$ and $\theta = 5$ and 10.

50%. Since the minimum travel time for the Wetstraat is about 1 min, whereas the overall ATWT for the green wave method is also about 1 min and for SOTL about half of that, the improvement in the average total travel times would be of about 25%, i.e., cars under a green wave method would take 33% more time to reach their destination than those under SOTL. This shows with a realistic simulation that SOTL greatly improves traffic flow compared to the current green wave method.

3.6 Discussion

The green wave method works well for regulating traffic on the Wetstraat, since most of the traffic flows through it. Still, having no concern as to the actual state of the traffic has several drawbacks. It can give a green light to a side street even if there are no cars on it or when a group of cars is about to cross in the other direction. Also, if the traffic density is high, the speed of the cars will be slower than that of the green wave. Furthermore, when a car misses a green wave, it has to wait a full cycle to get into the next one.

Having actual information about the traffic state enables SOTL to adapt to the current situation: it only gives green lights on demand, so time is not wasted for streets without cars, whereas streets with more cars, which thus have more demand, have more green lights. Cars do have to wait behind red lights, but since while doing so they are demanding to cross, it is very unlikely that a car will have to wait more than

$\phi_{\min}$. Moreover, when a car is stopped, a platoon is likely to be formed, accelerating the switching of green lights.

Another advantage of platoons is that they reduce entropy in the city, defined via the probability of finding a car in any part of the city. If there is maximal entropy, there is the same probability of finding a car anywhere in the city. This increases the probability of interference, i.e., that two cars will meet at an intersection, thus requiring one to stop. The opposite extreme is less desirable: if we have a certainty of the position of every car, it is because they are stopped, i.e., in a traffic jam. However, platoons offer a useful balance: there is a high probability that a car will be close to another car, i.e., in a group. Thus, there are many free spaces left between platoons, which other platoons can exploit to cross without interference. There will be interferences, but these will be minimal.

3.7 Future Work

The following list summarizes future work.

- A method similar to SOTL has been used successfully in the United Kingdom for some time, but only for isolated intersections (Vincent and Young 1986). Indeed, it is not obvious to expect that traffic lights without direct communication would be able to coordinate robustly. In any case, the technology to implement it is already available, so a pilot study could be quickly deployed in a real city. Since the traffic lights are adaptive, only a few intersections would have to be changed, to adapt to the control method used in the rest of the city. This also would make it easy to incrementally introduce them in large cities.
- We have observed that there is a monotonic relationship between the best θ and the traffic density (Cools 2006). Exploring this relation better could allow us to set a variable θ depending on the current traffic density measured by the traffic lights. However, since SOTL performs very well for a broad range of parameters, it does not require the calculation of precise parameters. In other words, SOTL is not sensitive to small changes in parameters, making it a robust method.
- The SOTL method could also be used to give preference to certain users, e.g., public transport or emergency vehicles. Simply, a weight would be given to each vehicle in the count κ_i, so that vehicles with preference would be able to trigger green lights by themselves. They would be equivalent to a platoon of cars, thus being seamlessly integrated into the system. This might be a considerable improvement compared to current methods, where some vehicles (e.g., buses in London, trams in Brussels) have preference and the rest of the users are neglected, in some cases even when there are no preferred vehicles nearby.
- The "optimal" sizes of platoons, depending on different features of a city, is an interesting topic to research. The parameters of SOTL can be regulated to promote platoons of a certain size, so knowing what size should be aimed at would facilitate the parameter search.

- It would be interesting to compare SOTL with the Dresden method (Helbing et al. 2005; Lämmer et al. 2006), which couples oscillators using self-organization, whereas SOTL has no internal phases or clocks.

3.8 Conclusions

In this chapter we presented results showing that a self-organizing traffic light control method considerably improves the traffic flow compared to the current green wave method, namely reducing average waiting times by half. These results are encouraging enough to continue refining and exploring similar traffic light controllers and to implement them in real cities, starting with pilot studies. However, we would not like to further motivate the use of cars with efficient traffic control, since this would increase traffic densities and pollution even more. Any city aiming at improving its traffic flow should promote in parallel alternative modes of transportation, such as cycling, walking, car pooling, and public transport.

Acknowledgments

We should like to thank the Ministerie van het Brussels Hoofdstedelijk Gewest for their support, providing the data for the Wetstraat.

References

Ashby, W. R. (1962). Principles of the self-organizing system. In Foerster, H. V., and G. W. Zopf, J., editors, *Principles of Self-Organization*, pages 255–278. Pergamon, Oxford.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2003). *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ.

Cools, S. B. (2006). A realistic simulation for self-organizing traffic lights. Unpublished BSc Thesis, Vrije Universiteit Brussel.

Federal Highway Administration (1998). *Traffic Control Systems Handbook*. U.S. Department of Transportation.

Gershenson, C. (2005). Self-organizing traffic lights. *Complex Systems*, 16(1):29–53.

Gershenson, C. (2006). A general methodology for designing self-organizing systems. Technical Report 2005-05, ECCO.

Gershenson, C., and Heylighen, F. (2003). When can we call a system self-organizing? In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J. T., and Ziegler, J., editors, *Advances in Artificial Life, 7th European Conference, ECAL 2003 LNAI 2801*, pages 606–614. Springer, Berlin.

Haken, H. (1981). Synergetics and the problem of self-organization. In Roth, G. and Schwegler, H., editors, *Self-Organizing Systems: An Interdisciplinary Approach*, pages 9–13. Campus Verlag, New York.

Helbing, D. (1997). *Verkehrsdynamik*. Springer, Berlin.

Helbing, D., Herrmann, H. J., Schreckenberg, M., and Wolf, D. E., editors (2000). *Traffic and Granular Flow '99: Social, Traffic, and Granular Dynamics*, Springer, Berlin.

Helbing, D., and Huberman, B. A. (1998). Coherent moving states in highway traffic. *Nature*, 396:738–740.

Helbing, D., Lämmer, S., and Lebacque, J.-P. (2005). Self-organized control of irregular or perturbed network traffic. In Deissenberg, C. and Hartl, R. F., editors, *Optimal Control and Dynamic Games*, pages 239–274. Springer, Dordrecht.

Heylighen, F. (2003). The science of self-organization and adaptivity. In Kiel, L. D., editor, *The Encyclopedia of Life Support Systems*. EOLSS, Oxford.

Heylighen, F., and Gershenson, C. (2003). The meaning of self-organization in computing. *IEEE Intelligent Systems*, pages 72–75.

Hunt, P. B., Robertson, D. I., Bretherton, R. D., and Winton, R. I. (1981). SCOOT-a traffic responsive method of coordinating signals. Technical report, TRRL.

Lämmer, S., Kori, H., Peters, K., and Helbing, D. (2006). Decentralised control of material or traffic flows in networks using phase-synchronisation. *Physica A*, 363(1):39–47.

Lendaris, G. G. (1964). On the definition of self-organizing systems. *Proceedings of the IEEE* 52(3):324–325.

Nicolis, G., and Prigogine, I. (1977). *Self-Organization in Non-Equilibrium Systems: From Dissipative Structures to Order Through Fluctuations*. Wiley, New York.

Porche, I., and Lafortune, S. (1998). Adaptive look-ahead optimization of traffic signals. *ITS Journal*, 4(3):209–254.

Prigogine, I., and Herman, R. (1971). *Kinetic Theory of Vehicular Traffic*. Elsevier, New York.

Schreckenberg, M., and Wolf, D. E., editors (1998). *Traffic and Granular Flow '97*, Springer, Singapore.

Shalizi, C. R. (2001). *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Wisconsin, Madison.

Skår, J. and Coveney, P. V., editors (2003). Self-Organization: The Quest for the Origin and Evolution of Structure. *Phil. Trans. R. Soc. Lond.* A 361(1807). *Proceedings of the 2002 Nobel Symposium on Self-Organization*.

Vincent, R. A., and Young, C. P. (1986). Self optimising traffic signal control using microprocessors - the TRRL MOVA strategy for isolated intersections. *Traffic Engineering and Control*, 27(7-8):385–387.

von Foerster, H. (1960). On self-organizing systems and their environments. In Yovitts, M. C., and Cameron, S., editors, *Self-Organizing Systems*, pages 31–50. Pergamon, New York .

Wiering, M., Vreeken, J., Veenen, J. V., and Koopman, A. (2004). Simulation and optimization of traffic in a city. In *IEEE Intelligent Vehicles Symposium (IV'04)*. IEEE pages 453–458.

Wolf, D. E., Schreckenberg, M., and Bachem, A., editors (1996). *Traffic and Granular Flow '95*, World Scientific, Singapore.

4

A Self-organizing Sensing System for Structural Health Monitoring of Aerospace Vehicles

N. Hoschke, C. J. Lewis, D. C. Price, D. A. Scott, V. Gerasimov, and P. Wang

4.1 Introduction

This chapter describes the development and operation of an experimental structural health monitoring system whose functionality is based on self-organization in a complex multiagent system. Self-organization within a system of many interacting components is generally understood to mean the formation of global patterns, or the production of coordinated global behaviours, solely from the interactions among the lower-level components of the system. The important characteristics are that the resulting patterns or behaviours occur at a larger scale than the individual system components and the interactions among the components are not influenced by a central controller or by reference to the emergent pattern or behaviour; they are purely local interactions. Self-organization in biological systems has been defined and discussed by Camazine et al. (2001), and Prokopenko et al. (2007) have discussed self-organization from an information-theoretic perspective.

The system that is described in this chapter consists of a large number ($\sim$200) of semiautonomous local sensing "agents," each of which can sense, process data, and communicate with its neighbours. In this context self-organization means that the agents will produce a system-level response to external events or damage that is entirely the result of local communication among the agents and is not influenced by a central controller or by any system-level design. The main benefits of this approach lie in scalability (the system performance is not limited by the computational and communication capability of a central controller) and in robustness (there is no single point of vulnerability, such as would be represented by a central controller).

4.1.1 The Requirements of Structural Health Monitoring

Structural Health Monitoring (SHM) is a new approach to monitoring the integrity and functionality of structures. It is expected to enhance, and ultimately replace, the traditional approach of periodic inspection for the maintenance of, e.g., aerospace and other transport vehicles, bridges, buildings, and other safety-critical infrastructures. SHM uses information from sensors permanently embedded in the structure to detect

events or conditions that may lead to damage and/or to detect damage at an early stage and monitor its development with time. Initially, SHM systems will be used to plan maintenance as required, rather than the current practice of maintenance at predetermined intervals, but ultimately it is likely to be used to manage and monitor materials and structures with self-repair capabilities.

SHM systems will be required to monitor very large numbers of sensors, to use the information deduced from the sensor data, to diagnose damaging situations and consequent damage, to form a prognosis of the future safety and functionality of the structure, and to initiate and monitor mitigation and repair procedures as required. Different forms of damage can develop on very different spatial and temporal scales (compare, e.g., the effects of a sudden major impact with those of slowly developing corrosion or fatigue), and the SHM system must be able to respond effectively in all cases.

Of paramount importance for SHM of safety-critical structures are the requirements for system robustness and scalability. The system must be capable of continuing to operate effectively in the presence of damage to itself and/or to the structure, and it must be able to operate efficiently, both locally and globally, even though it may be monitoring very large numbers of sensors. These requirements mitigate against the use of traditional centrally controlled engineered systems, with their inherent points of vulnerability, in favour of distributed adaptive systems.

4.1.2 The Approach to SHM System Development

CSIRO, with support from NASA, has been developing an experimental structural health-monitoring concept demonstrator (CD) test-bed system for the detection of high-velocity impacts, such as may occur due to the impact of a micrometeoroid on a space vehicle. The distinguishing feature of this system is that its architecture is based on a complex multiagent system and its behaviours and responses are developed through self-organization. It has no central controller. This approach endows the system with a high degree of robustness, adaptability, and scalability.

The test bed has been built as a tool for research into sensor design, sensing strategies, communication protocols, and distributed processing using multiagent systems. Each of the semiautonomous sensing agents contains a suite of sensors to enable it to gather data related to the state of the structure (generally, but not necessarily, referring to the agent's local region) and to provide a facility to perform some processing of these data and the ability to communicate with neighbouring agents. These agents may also have the capability to engage in active tasks (e.g., repair functions), or there may be other agents to perform these functions. Agents may be embedded in the structure, eventually being integrated into the materials, or they may be mobile and free to roam regions of the structure.

A number of recent articles have described the development of the hardware and software of the basic CD system of embedded piezoelectric sensors and processing electronics distributed throughout the structure, along with some multiagent algorithms to characterize impacts and subsequent damage, see, e.g., Price et al. (2004), and Scott et al. (2005), Hoschke et al. (2006), and Prokopenko et al. (2006). The CD system

has been designed to be highly flexible: by replacing the sensors and their associated interface and data acquisition electronics, the system can be readily reconfigured for other applications.

This chapter provides more detail on the system described in our earlier publications (see references in previous paragraph), with an emphasis on the communications among agents, the development of the gradient field algorithm for this system, and the incorporation of a robotic mobile agent which can move over the surface of the CD skin as an independent agent of the self-organizing system. This mobile agent can carry out sensing functions that the embedded agents cannot, and it is able to communicate with embedded agents of the CD structure in its vicinity. It is envisaged as the forerunner of, eventually, a swarm of small robots that will cooperatively perform both inspection and repair functions. The essential point is that the functions and behaviours of the mobile agent (or agents) are determined by self-organization of the entire system of fixed and mobile sensing agents.

4.1.3 Overview of the Experimental System Operation

The CD system consists of a basic structure to be monitored, which is a rigid framework in the form of a hexagonal prism ($\sim$ 1 m in height and $\sim$ 1 m across the hexagonal section) covered by an aluminium skin 1 mm thick (Fig. 4.1). Bonded to the inner surface of the aluminium are 768 small (2.5 mm diameter) piezoelectric sensors in groups of four, in the form of a regular array. A block of electronics that constitutes an agent of the system is connected to each group of four sensors. There are 192 such agents,

Fig. 4.1. The hexagonal prism physical implementation of the CD test-bed structure, lying on its side with the end open to reveal the cellular internal structure of the electronics.

distributed in a square array with 32 on each of the hexagonal faces of the structure. Each agent monitors its group of four sensors, acquires and analyzes the data they produce, and communicates with its four neighbours. Figure 4.1 is a photograph of this structure, showing the array of agents covering the inner surface. An agent, together with the region of the skin that it monitors, is referred to as a cell, though the terms "agent" and "cell" are used interchangeably throughout this chapter.

An impact on the surface of the aluminium skin excites elastic waves, which are detected by the piezoelectric sensors. High-velocity impacts are simulated using short laser pulses and/or steel spheres fired using a light-gas gun. Repeatable low-velocity impacts are produced using a pendulum. The agents that detect the impact also locate its position and estimate the severity of the damage from the sensor signals.

A number of multiagent algorithms have been developed (Price et al. 2004; Scott et al. 2005; Hoschke et al. 2006; Prokopenko et al. 2005b, 2006) to identify impacts, the extent of the damage caused, and networks of multiple impacts. The state of the system at any time is monitored on an external computer (the system visualizer), which acts as another agent of the system that requests and displays state information from the embedded agents. The visualizer does not have any control function; it simply monitors and displays the states of the other agents.

A robot has recently been developed to move around the outer surface of the skin to provide a mobile sensing capability. At this stage it carries a video camera to record visual images of damage, but it could carry other sensors as well. This robot can communicate through the aluminium skin, using ultrasonic signals, to the agent in its immediate vicinity. It is guided towards damage sites by information it obtains from the embedded agents as it moves. The necessary information to guide the robot is produced collectively by the multiagent system—a self-organized response to the detection of an impact by a local agent. The robotic agent is not controlled centrally; rather it navigates purely via the local information it obtains from the agents in the underlying structure as it passes by, so the robotic movement is a self-organized response of the system of agents to the impact. This agent-based response of the robot is robust, scalable, and adaptable, similar to the agent-based response of the system to impact detection, and the overarching principle of the project.

4.1.4 Structure of the Chapter

The next sections contain more detailed descriptions of the two major components of the system, the fixed structure with its embedded agents and the mobile robotic agent. Section 4.2 describes the fixed structure of the CD and its embedded sensing agents, including details about the sensors, the impacts and their diagnosis, the development of self-organizing algorithms for robot guidance, and details about the ultrasonic signals used by the embedded agents to communicate with the robotic agent. Section 4.3 describes the robotic agent—its hardware, its modes of motion, and the ultrasonic transducers it uses for communication with the fixed agents on the CD structure. The communications protocol is outlined in Section 4.3.4, followed by a description of the way in which the communications with the embedded agents are used by the robot for stepping and navigation. The intended future use of a video camera by the robot

for damage inspection is outlined. These sections are followed by a short description of the system visualizer (Section 4.4), a summary of what has been achieved, and an indication of the next steps in this development program.

4.2 CD Embedded System Components: Hardware and Software

4.2.1 CD Architecture and Hardware

The initial goal of the test bed is to detect and characterize impacts on the skin, as well as to diagnose to the accumulated damage. The skin consists of 48 aluminium panels (eight on each side of the hexagon), each of which contains four "cells" (Fig. 4.2). Cells are the fundamental building blocks of the system—the electronic modules that contain the sensing, processing, and communication electronics. Each cell is an agent of the distributed multiagent system, and communicates with its four immediate neighbours.

Each cell occupies an area of $\sim 100\,\mathrm{mm} \times 100\,\mathrm{mm}$ of the skin, mounted on the inside of which are four piezoelectric polymer (PVDF) sensors to detect the acoustic waves that propagate through the skin as a result of an impact. Thus the complete test bed contains 192 cells. One of the panels and its four cells are shown in Fig. 4.2.

The cell electronics are constructed as two submodules mounted directly on top of each other. One of the submodules, called the network application submodule (NAS), contains the communications and processing hardware, while the data acquisition submodule (DAS) contains the analogue electronics and digitization hardware specific to

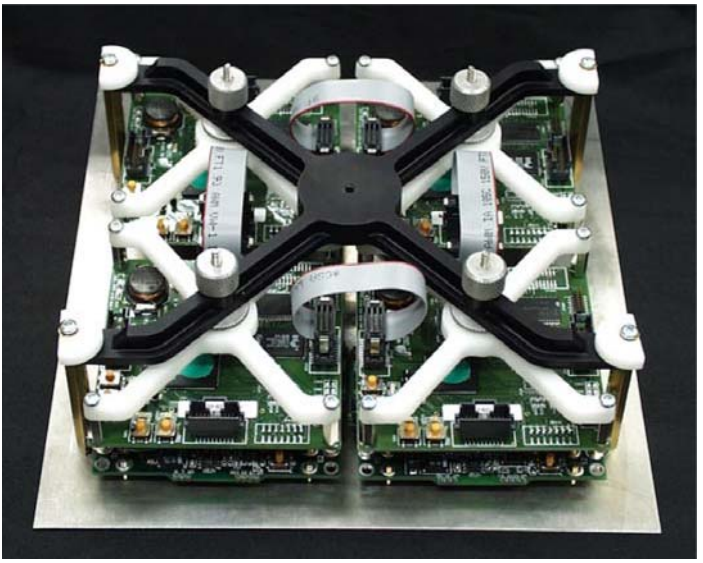


Fig. 4.2. Aluminium panel containing four cells. Each cell consists of a data acquisition submodule (DAS) below a network application submodule (NAS). Each cell is connected to its four immediate neighbours, via the ribbon cables that can be seen in the photograph, to form a square network array.

the attached sensors. A benefit of this division is that the NAS is flexible enough for almost any SHM sensor network application, and only the DAS needs to be changed to accommodate the different sensors that may be required in different applications. Further details of the electronics can be found in Hedley et al. (2003).

4.2.2 Piezoelectric Sensors

The aluminium panels that form the CD skin have the dimensions $200\,\mathrm{mm} \times 220\,\mathrm{mm} \times 1\,\mathrm{mm}$. Each panel has an array of piezoelectric sensors bonded to one side. This array consists of sixteen polyvinylidene fluoride (PVDF) discs (110 μm thick, 2.5 mm in diameter, coated on one side with silver ink, while the other side is bonded to the aluminium) and four lead zirconate titanate (PZT) discs (0.5 mm thick, 2.5 mm in diameter, with fired-on silver electrodes on the two flat faces). These are bonded to the panels in the arrangement shown in Fig. 4.3. The sensors that detect the elastic waves caused by impacts were made from PVDF, as this piezoelectric material has high sensitivity as a receiver (but is a relatively poor transmitter), is inexpensive, is relatively easy to fabricate into transducers, and would not significantly mass load the surface. The single transducer at the centre of each cell, designed primarily as a transmitter to communicate with the robot and, at a later stage, to generate signals for damage evaluation, is made from PZT, which is a more efficient transmitter than PVDF. In the fully populated demonstrator there are 48 panels, and therefore 768 PVDF and 192 PZT transducers.

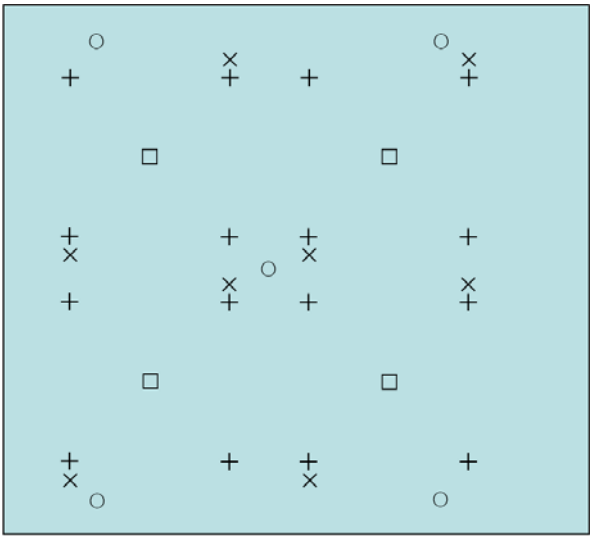


Fig. 4.3. Location of sensors on aluminium panel: + PVDF transducers; □ PZT transducers; ○ holes for attachment of electronics; × small divots for precise location of DAS boards.

The initial role of the PVDF sensors was simply to enable the detection, characterization, and location of impacts on the aluminium skin. Each group of four PVDF sensors in each cell would detect elastic waves resulting from an impact travelling through the aluminium plate, and the agent would use this information to determine the location and severity of the impact. This role has now been expanded, and these sensors also act as the receivers of communication signals from the mobile robotic agent.

The PZT transducers, located in the centre of each group of four PVDF sensors, fulfil a number of functions. They can be used as transmitters to send ultrasonic waves through the panel for subsequent detection by the PVDF sensors before and after impacts on the panels. This allows firstly the calibration of the PVDF sensors (or at least a measurement of their sensitivity), and secondly the possible determination of the state of damage of the panel after the impact has occurred. The PZT transducer in each cell is also used to transmit communications from the embedded agent to the mobile agent when it is positioned on the skin in the region of that cell.

4.2.3 Impacts and Simulated Damage

As outlined above, the CD's present function is to detect impacts and diagnose the level of damage to its operation. The software system has been designed to distinguish between hard impacts (those with a high impulsive force) resulting in damage to the panel (a critical impact, requiring attention by the robot, or 'mobile agent'), lesser impacts (low impulsive force) resulting in damage that does not need immediate repair (a noncritical impact), and electronics and/or communications failures not caused by an impact. During system development so far, real damage has been avoided, so that costly panels would not have to be replaced. The simulation of the two levels of damage has been done by using a pendulum to strike the panel with either a high impulsive force (for a 'critical' or 'hard' impact), or a low impulsive force (for a 'noncritical' or 'soft' impact). Neither of these types of impact causes damage to the panel, but they do generate elastic waves in the panel of higher or lower amplitude, respectively. By manually attaching a red marker to the cell that has suffered a hard impact (or a green marker for a low impact), a visual difference between critically and noncritically 'damaged' cells is provided. This allows the secondary inspection system (using the robot with a small video camera and simple frame-grabbing software to determine the colour of the marker) to visually distinguish between the consequences of hard and soft impacts by colour recognition rather than by visually detecting real damage such as a penetration of the panel. The location and diagnosis of the damage by this secondary inspection technique can then be compared with estimates made from the piezoelectric sensor data.

During system development and testing, repeatable 'hard' and 'soft' impacts were applied by using a pendulum apparatus, which could be set to deliver any impulsive force repeatedly and reliably, only limited by the highest potential (and hence kinetic) energy obtainable from the pendulum. The apparatus was held against a panel (using

three felt-covered stand-offs), and the pendulum was drawn back to the desired and repeatable height for striking the panel with the selected impulsive force.

Subsequent to such an impact, the PVDF sensor data was used to estimate the position (using the triangulation method outlined in Prokopenko et al. 2006) and severity (using self-organized maps, as described in the next subsection) of the measured impact.

4.2.4 Impact Signals and Sensor-based Diagnosis: Use of Self-organized Maps (SOMs)

A general discussion of the approach to damage diagnosis by self-organization is given in Price et al. (2004). In this case self-organizing maps (Kohonen maps; Kohonen 2001, 2003) have been implemented to classify impact severity, distinguishing critical impacts, which have ruptured the skin, from noncritical impacts. A previous discussion of the application of SOMs to the analysis of impact data is given in Prokopenko et al. (2005b). Electronic failures, which are detected when a cell loses its communication capability, are distinguished from critical impacts that have damaged the electronics by the absence of an impact recorded by a neighbouring cell.

The aim of this signal-based diagnosis is to identify high- and low-severity impacts in different regions of a panel: specifically, whether an impact has occurred within the cell that has recorded the signals or within one of the other three cells of the panel. In general, a cell's sensors will detect an impact that occurs anywhere on the panel on which the cell is located, but usually not if the impact occurs on another panel. The diagnosis should be able to unequivocally identify the cell on which the impact occurred (even if that cell has been damaged to the extent that it can no longer communicate), an approximate position within that cell, and whether the impact was of high or low severity.

Training of the self-organized maps was done on a single panel, using hard and soft impacts produced by the pendulum apparatus at a number of locations within each of the four cells on the panel.

The initial aim was to use the SOM to identify the following four conditions:

- A soft impact occurred within the cell.
- A hard impact occurred within the cell.
- A soft impact occurred outside the cell.
- A hard impact occurred outside the cell.

If this diagnosis can be made for each cell on a panel that has suffered an impact, then the cell on which the impact occurred can be identified unambiguously.

For a particular cell on the panel, 100 soft and 100 hard impacts at random positions within the area of the cell were sampled. These samples covered most of the cell's area thoroughly, giving roughly three impacts per square centimetre within the cell. Further, for impacts outside the cell, 100 soft and 100 hard impacts were sampled from the areas of the three remaining cells.

An impact event is recognized when a sensor signal threshold is exceeded. The cell's DAS board then acquires 256 samples from each of its four sensors. It cannot be

assumed that any of the signals reflect an accurate time of arrival of the impact pulse because of the use of a constant threshold. In order to reduce the memory requirement for each SOM and maximize the size of the SOM array that can be stored, the string length was reduced to 64 by four-point subsampling of each 256-sample signal. A data input vector used for training the SOM consists of the 64-point data string from each of its four sensors in the relevant cell. This allowed a 10×10 SOM array to be stored in binary format in 50 kB of flash memory on each agent.

For the purpose of forming the data vectors, the four sensors were ordered based on time of arrival of the signal (because the training set is taken for one particular orientation and the cells on the CD are not always in the same orientation). This makes the SOM orientation independent, but at the expense of geometric information about the direction of the position at which the impact occurred.

In order to improve the efficiency and effectiveness of the learning process, the 10×10 SOM was trained in an unconventional way. The 10×10 array was divided into four 5×5 arrays, with each of these smaller SOMs assigned to learning only one of the four conditions listed above that is to be identified. Each of the 5×5 blocks was then trained separately using the subset of the training signals that corresponded to the particular block, e.g., the 5×5 block assigned to soft impacts within the sensing cell was trained using only the signals produced by soft impacts within the sensing cell. The boundaries of the 5×5 blocks were wrapped around (top to bottom, left to right) to provide continuity at the edges. The four 5×5 blocks were then assembled into a single 10×10 SOM array. While the conventional approach to SOM formation is purely unsupervised, data-driven learning, the present approach may be considered to be adding an element of supervision to the learning process.

Training of each of the four 5×5 SOMs was controlled by the following parameters:

- The neighbourhood function is a constant over the 5×5 block, independent of location in the array and training iteration number. This was a simplifying assumption resulting from the small size of the blocks.
- The maximum number of training iterations (epochs) was $T = 2500$.
- The learning rate function $\eta(t)$, which is required to decrease with increasing training iteration number, was taken as

$$\eta(t) = 0.10 \cdot e^{(-t/T)}.$$

The trained vectors thus produced were stored as a SOM on each of the agents on the CD. When identifying the type of impact, the four smaller SOMs are evaluated as a single 10×10 SOM.

In order to evaluate the accuracy with which signals can be identified with the resulting SOM, only half of the training signals were used to train the SOM and the other half were used to evaluate the accuracy of recall and precision. Several separate runs showed a consistent accuracy of $\sim 93 \pm 1\%$, independent of the signal trigger threshold over a range from $\sim 1.2\%$ to $\sim 3.6\%$ of the maximum signal amplitude. Given that each impact is detected by four cells, the probability of an incorrect assignment of an impact location is very small.

For greater accuracy, the SOM on each agent could have been trained separately, which would have individualized the SOMs to take account of the inevitably different sensitivities of the sensors in different cells. Ultimately, it is expected that SOMs will learn on-line, so even if they are initially trained with a common data set, the subsequent learning will develop a set of individualized SOMs.

So far the SOM-based impact diagnoses have proved to be highly reliable, but tests to date have all been carried out using the same impact mechanism with which they were trained. It remains to be seen whether the SOMs will retain this high accuracy with impacts of different origin but similar spectral characteristics.

4.2.5 Self-organized Robot Guidance: Gradient Field Algorithms

There are two related methods by which navigation of the robot can be achieved, both directed by self-organization. Firstly, algorithms based on ant colony optimization (ACO) (Dorigo and Di Caro 1999; Dorigo and Stützle 2004; Prokopenko et al. 2005a) have been developed in earlier work to link subcritical impact locations by a simulated pheromone trail and a dead-reckoning scheme (DRS) that form a minimum spanning tree (Prokopenko et al. 2005a). The decentralized ACO-DRS algorithm has low communication cost, is robust to information loss within any individual cells, and allows navigation around critically damaged regions in which communication capability has been lost.

An alternative scheme evaluated by Prokopenko et al. (2005a) is a distributed dynamic programming algorithm, employing a gradient field (GF) set by each impact cell.

Although the concept of the robot following the self-organized pheromone trails produced by ACO is appealing, there is a trade-off between the low communication cost of the ACO-DRS algorithm and a better quality of the minimum spanning tree approximation computed by the gradient-based algorithm (Prokopenko et al. 2005a). The GF algorithm also ensures that each cell in the system has a valid gradient value: the ACO-DRS algorithm does not guarantee a pheromone value in every cell.

The approach that has been implemented is the GF algorithm. The basic principle of gradient propagation is very simple. All cells are initiated with a high integer value for the gradient (255). When a cell detects an impact, its gradient value is set to zero. At each time step, each cell compares its gradient value with that of each of its neighbours. If a cell finds a neighbour with a gradient value lower than its own, it takes this value plus one as its new gradient value. This is repeated at subsequent time steps until a stable gradient field is produced over the whole array of cells; i.e., the gradient produced by the impact propagates throughout the system of agents. It is independent of the number of neighbours each cell has, so it is robust in the presence of failed cells. Multiple impacts produce a gradient field with multiple minima, analogous to a topographic map of a surface with minima that correspond to impact sites or the surface of a trampoline that has a number of separated weights on it.

Figure 4.4 illustrates a grid of 23×15 simulated cells, each containing a number representing the value of the gradient field (at that cell position) that has resulted from three impacts (denoted by black cells). The positions of two robots are denoted by

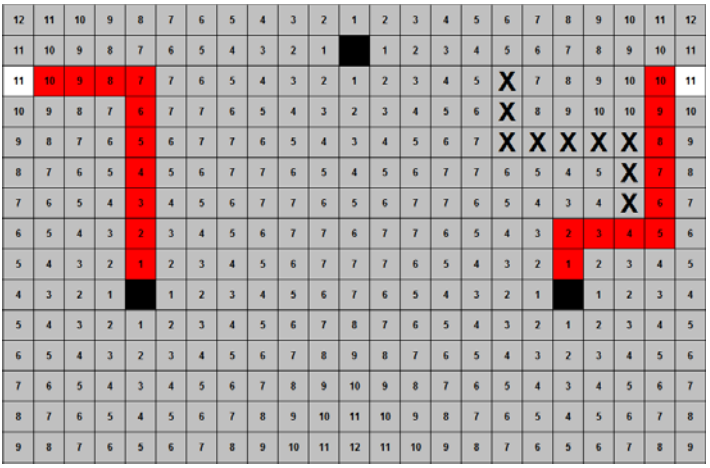


Fig. 4.4. A grid of simulated cells showing the value of the gradient field in each cell resulting from three impacts (black cells). The initial positions of two robots are shown with white squares. Crosses denote cells that are not operational. The paths taken by the robots to the impact sites are shown in dark grey.

the two white cells, and the shortest paths found by the robots from these positions to the impact sites are shown in dark grey. Nonoperational cells that are unavailable for the robot to traverse are denoted by crosses. As each damaged cell is attended to, the gradient field will be updated and the robot(s) will move to the remaining damaged site(s).

The modification to the gradient field (when the damage is repaired or inspection shows that it can be ignored) is achieved as follows. The previously impacted cell resets its impact flag and then increases its gradient to a value that is one greater than that of its neighbour with the lowest gradient value. All cells then check all their neighbours. If a cell has a gradient value lower than those of all its neighbours, and it has not been impacted, its gradient value is incremented by one. The new gradient values reach equilibrium in less time than it takes the robot to perform one step. Repetition of this algorithm removes the minimum associated with the repaired cell. This action is analogous to one of several separated weights being removed from the surface of a trampoline.

To allow the robot to identify and respond to different classes of events, a gradient field could be set up with minima of varying depths corresponding to the severity/importance of each event. However, steps would have to be taken to avoid the complication of the robot being attracted to nearby minima in preference to more distant, but perhaps more important, minima. A simple solution to this problem is to model different classes of events on separate gradient fields and set all minima on a single gradient field to be equally important (deep). The multiple classes of gradient field may all model different routes to their respective sites of importance. This approach means that the robot makes the decisions on which gradient to follow. One way to process this gradient information is for the robot to respond to events in order

of priority by exhausting one class of gradients before switching to a gradient class with lower priority, and so on. It should be noted that each time the robot puts its foot onto the panel, the value of the gradient field for each type of damage is communicated to it. If, for example, the robot is following a lower-priority gradient field such as the 'noncritical impacts' field and a critical impact occurs, the appearance of the updated critical impact field values will cause the robot to follow this higher priority field on its next and subsequent steps.

This multiple gradient field solution has been implemented and currently the robot responds to four classes of gradient/event:

- High-severity impacts, representing critical damage—perhaps penetrating impacts and/or cell destruction.
- Low-severity impacts, representing noncritical damage—nonpenetrating impacts and damage which does not affect system behaviour.
- Damage not caused by an impact, such as communications and/or electronic failure.
- Dock: models the shortest route to the robot's docking station for recharging, downtime, etc.

The robot can adopt different criteria to prioritize the order in which it visits impact sites. A possible modification to the behaviour outlined above would be to allow the robot to visit sites of lesser importance if they are on or near the intended path to a site of greater importance.

Major benefits of this algorithm are its stability and its fast dynamic response. A mobile agent in the system can determine the direction of the shortest path to the nearest impact location through interrogation of the local cell group. This is analogous to a ball on a slope; the ball need not know where the bottom of the hill is to know which way to roll—simply knowing the gradient of its local piece of hill is sufficient. In the GF algorithm, the shortest distance to an impact location can be determined simply from the value of the gradient in a particular cell, since each cell increments the lowest gradient value of its group of neighbours by one unit. Note that this algorithm, with a separate gradient field for each type of damage, does not suffer the 'local minima problem'. Within each gradient field each damage site has equal priority, with the same 'depth' or field value, damage of different priority is represented by values contained in different fields.

Figure 4.5 shows a real gradient field produced on the CD as a result of two noncritical impacts, located in the black cells. The gradient value at each cell is indicated by the shade of grey. White cells are those with which the robot cannot communicate. This may be due to electronic failure or, as is the case here, sensors have not yet been fitted.

4.2.6 Communications with the Robot: Distinguishing Impact and Communication Signals

Communication between an embedded agent (cell) and the robot utilizes ultrasonic signals propagated through the aluminium skin of the cell. The robot transmits and

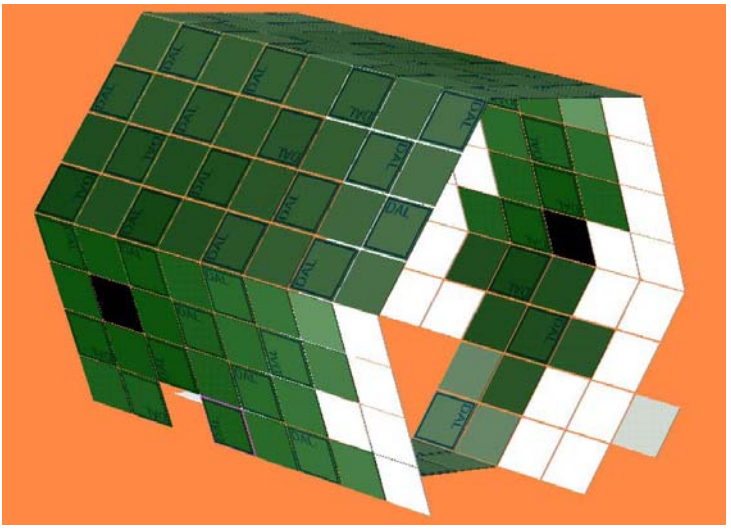


Fig. 4.5. An image of the gradient produced by two noncritical impacts that occurred at the black cells. The squares indicate the cells on the surface of the hexagonal prism testbed. The gradient values are shown as shades of grey, with a lower gradient being darker. The white cells are those with which the robot cannot communicate. Absent cells in the image are those that have been physically removed from the CD.

receives ultrasonic signals using transducers mounted in the centre of each foot. Further details about the robot's transducers and the communications sequences are given in the next section. The agent transmits via the PZT element in the centre of the cell and receives signals through one of the four PVDF elements that are used for impact detection.

A communication is initiated when the robot places one of its feet on the region of skin monitored by agent A, say. In order to initiate a communication sequence, the robot then transmits a tone burst from the ultrasonic transducer in this foot (see Sec. 4.3), which consists of five cycles at a driving frequency of 400 kHz. This corresponds approximately to the lowest-order radial mode of the robot transducer disc, and it excites the zeroth-order antisymmetric (A_0) guided elastic wave mode of the aluminium plate (Rose 1999). The agent A distinguishes this signal from that due to an impact on the basis of its spectral content.

A five-cycle tone burst has a spectral width of $\sim 20\%$ of the centre frequency, and this will be increased by the spectral response of the transmitting and receiving transducers. Nevertheless, it is expected to have significantly different spectral characteristics to an impact-generated signal. At this stage the agents use a simple combination of the responses of two band-pass filters with different cut-off frequencies to distinguish impact events from communications events, but more sophisticated processing could be implemented readily.

A communications sequence is completed when the agent receives a specific acknowledgement signal from the robot. The issue of an impact that occurs during

a communications sequence has not yet been dealt with; at this stage it may result in a corrupted communications packet, but will not otherwise be recorded. This is not an urgent issue for the present system, since the robot foot would shield the cell during a communications sequence, though an impact might damage the robot. However, it raises potential issues of impact damage to the robot, as well as to the cell when much smaller robots are in use.

More details of the cell-robot communications are given in the next section.

4.3 The Mobile Robotic Agent

4.3.1 Development of the Mobile Agent

An important feature of the CD system is an ability to support mobile (robotic) agents that can roam the exterior surface of the test bed, communicating with the fixed agents embedded in the underlying structure. The function and operation of such an agent is described in this section, and it should be emphasized that it is not controlled centrally, but rather cooperatively with the network of fixed local agents with which it communicates.

It should also be emphasized that the system described here is no more than a test bed, whose primary purpose is to investigate the practicality of the self-organized complex system approach to damage diagnosis and response. Thus, details of the specific hardware implementation (such as the use of air suction for the robot's attachment, which is obviously inconsistent with a space-based application) are not considered to be important at this stage. While the present implementation of the robot is bulky and represents a single point of failure, the eventual aim is to develop a swarm of very small robots that can perform internal or external tasks cooperatively. The work described in this chapter represents a first step towards that ultimate goal.

Why is a robotic agent needed in an SHM system? When sensing impacts using passive sensors, the information received may be insufficient to characterize the damage, and where damage is detected it may need to be repaired. One approach to obtaining additional damage data and to providing a crude repair capability is the development of a mobile robot that can move around the outside skin (Fig. 4.6).

The robot moves rather like an inchworm, with its design based on an articulated box section with six degrees of freedom. The joints are driven by commercial model aircraft servos and have no position feedback to the controlling processor. The robot is equipped with six suction cups on each of its two feet, and a pneumatic system with a variable speed vacuum pump and electrically controlled valves that allow it to selectively attach and detach its feet to and from the surface. To allow the robot to find the surface and attach to it reliably, there are two optical range finders on each foot that measure the distance to the surface and the angle of the surface. A lithium polymer battery supplies power to the robot for approximately 30 min of operation before recharging is necessary.

The robot has two modes of locomotion. The first is very much like an inchworm, as mentioned above: to move forward the robot alternately stretches out and contracts

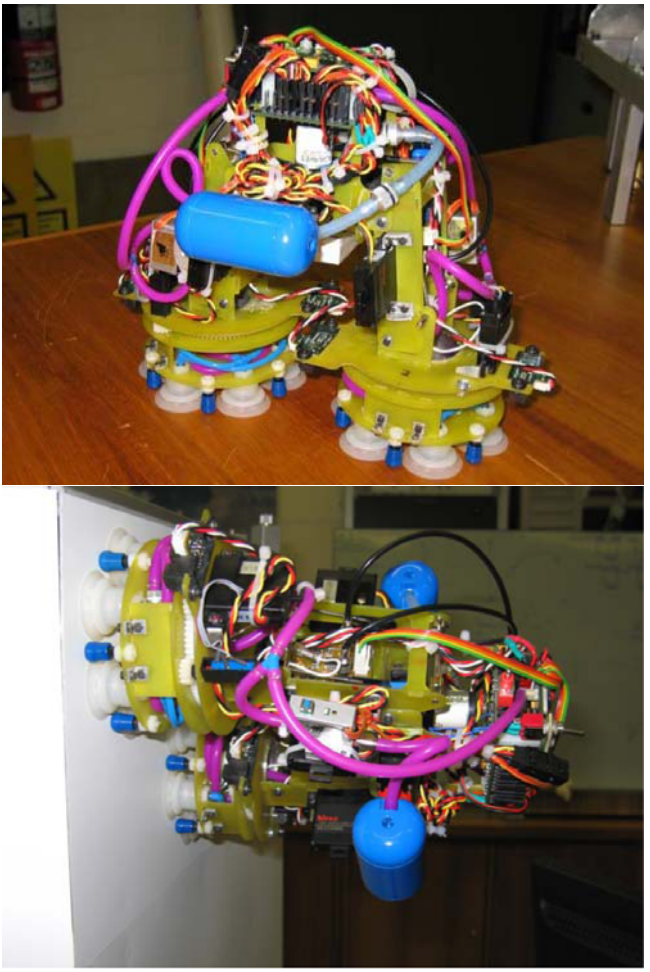


Fig. 4.6. The robot on a horizontal bench (upper), and on a vertical face of the test bed (lower).

whilst detaching and attaching its feet in sequence. The second mode requires the robot to detach one foot, pivot $180°$ around the other (attached) foot and then reattach the first. It can change direction by pivoting through any angle up to $360°$. Initially the robot will carry two small video cameras, one on each foot (Fig. 4.7), which will send images back to the network for further analysis. In future other sensors may be included, such as an active ultrasonic transducer that can interact with the piezoelectric receivers embedded in the skin for ultrasonic damage evaluation.

The robot communicates with the fixed agents in the network using piezoceramic (PZT) ultrasonic transducers in both feet (Fig. 4.8) to pass messages through the aluminium skin to the underlying cells. The development and performance of these transducers is described below. The fixed agents receive messages via one of the four

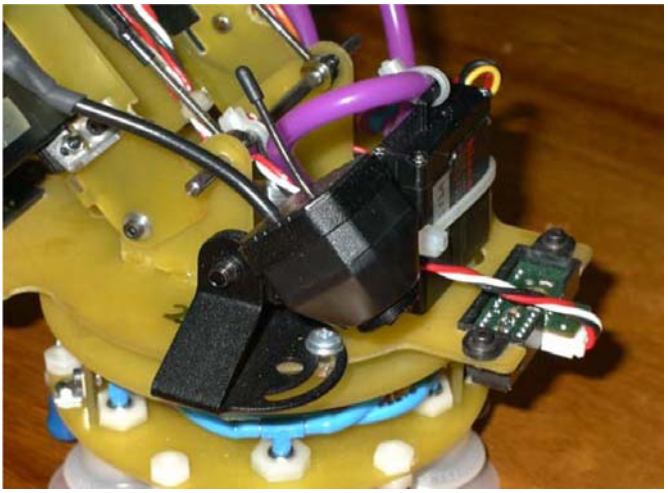


Fig. 4.7. Close-up of one foot of the robot, showing the inspection video camera (foreground), and one of the optical range finders (right).

piezoelectric polymer sensors that are used for detecting impacts. A fifth transducer, in this case a piezoceramic, has been added at the centre of each cell for transmission of messages from the cell to the robot.

4.3.2 Ultrasonic Transducer

An ultrasonic transducer is mounted on both of the robot's feet for communicating with the cell on which it is placed and indicating its foot position to the cell.

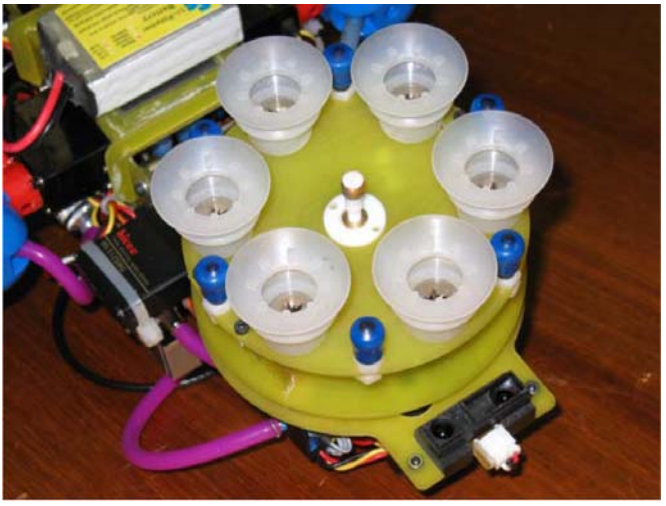


Fig. 4.8. The ultrasonic transducer mounted in the centre of the robot's foot.

The transducer's active element is a 5-mm-diameter, 2-mm-thick PZT ceramic disc, the outer surface of which is covered by a bonded wear plate of alumina (aluminium oxide), to protect the transducer from abrasion from the aluminium skin of the CD. The transducer housing is spring-loaded and inserted into a threaded rod, with two thumbscrews on the outside enabling the position of the transducer with respect to the footplate of the robot to be varied. This construction allows variation of the force with which the transducer is pressed against the CD skin, in order to maximize the ultrasonic coupling and minimize wear. Note that the design aim was to use dry coupling, i.e., to couple this transducer to the aluminium skin without any fluid couplant.

This transducer has its lowest-order thickness resonance just under 1 MHz (at 960 kHz). The lowest-order radial resonance of the disc occurs at 460 kHz, and is reduced to $\sim$ 400 kHz when bonded into the transducer housing.

In order to maintain efficient transmission of ultrasound between the transducer and the aluminium skin, it is important to ensure that the transducer face sits flat on the aluminium surface. While the robot's foot has been designed to try to assist this—by the inclusion of stops between the (flexible) suction pads (as can be seen in Fig. 4.8), which are pulled onto the aluminium skin by the applied vacuum—gravity, slight differences in the suction force at each pad, and surface irregularities on the aluminium may all contribute to preventing this from happening. Consequently, to ensure maximum coupling, a flexible head for the transducer has been developed using a ball-and-socket coupling mechanism supported by a silicone rubber boot bonded to both ends of the interconnecting parts. This head has been designed to give an angular variation of at most 4° in any direction with respect to the axis of the transducer (Fig. 4.9). Limiting this range is important in order to prevent any damage to the connections to the transducer's electrodes; the ball-and-socket construction and the flexible boot are designed to achieve this. This has proved to be a simple, inexpensive, and highly effective method of ensuring good coupling between the transducer face and

Fig. 4.9. Robot transducer mounted on a flexible head.

the aluminium. Further details about the development, construction, and performance of this transducer will be reported elsewhere.

4.3.3 Communication with the Embedded Agents

The 2.5-mm-diameter, 0.5-mm-thick PZT elements bonded to the inside of the aluminium panels are used as transmitters, both for communicating with the robot and, in principle, for active ultrasonic evaluation of damage. (A problem with the original design of the DAS boards prevents their use as ultrasonic receivers.) These elements have their lowest frequency resonance near 1 MHz, which is the fundamental radial mode of the disc. Their other major resonance is the first-order through-thickness resonance at about 4 MHz. In between these two are various higher-order radial modes, but the separation of the first radial and through-thickness modes is sufficient to prevent significant mixing of these modes. The 1-MHz radial resonance, with a wavelength equal to the disc diameter ($\sim$ 2.5 mm), couples efficiently with the large in-plane component of the A_0 waveguide mode of the aluminium panel at this wavelength and frequency combination.

The electronics of the DAS board has low-pass filters on the receiver channels to attenuate any frequency components above 1.55 MHz. This is necessary because the analogue-to-digital converters on the Texas Instruments TMS320F2812 DSP chips that control the DAS are operated at a sampling frequency of 12.5 MHz. (They have a maximum sampling rate of 16.6 MHz, but other constraints prevented operation at this rate.) This allows each of the four channels (one for each of the four PVDF transducers on each cell) to be sampled at 3.125 MHz, which means that the highest frequency that can be sampled without aliasing is about 1.56 MHz (the Nyquist frequency for this sampling rate).

For these reasons the A_0 guided mode of the aluminium panel at $\sim$1 MHz, with a phase velocity of $\sim$ 2.5 mm/μs and substantial in-plane and out-of-plane displacement components, was chosen as the communications channel.

The robot transducer has its lowest-order thickness resonance at 960 kHz, and this couples well into the panel A_0 mode for communications.

The lowest-order radial resonance of the robot's transducer occurs at $\sim$ 400 kHz. This resonance also couples efficiently into the A_0 plate mode, which has a lower phase velocity ($\sim$ 1.75 mm/μs) at this frequency and has been used for generating the signals that initiate a communications sequence and enable the position of the robot foot to be determined by triangulation (see below).

The sequence of events for a communication between the robot and an embedded cell/agent is described in the Table 4.1.

The times of arrival of the tone-burst signal at the agent's four PVDF sensors, which are required for triangulation as outlined in Step 4 in Table 4.1, are calculated by cross-correlation with a binary 400-kHz tone-burst filter. The peaks in the cross-correlations give the arrival times at the sensors. Triangulation is then done using a look-up table, as described by Scott et al. (2005) and Prokopenko et al. (2006).

The ultrasonic communications signals employ a 967.7-kHz carrier signal introduced into the aluminium skin. When the robot is transmitting, the carrier is generated by the robot's transducer and is received by the four PVDF impact sensors in the cell

Table 4.1. Sequence of communication events.

Sequence	Embedded agent/cell action	Robot action
1	Agent awaits a detectable event.	Robot moves foot to cell location.
2		When robot foot is attached, the robot transmits a tone burst (five cycles at 400 kHz) to initiate communications.
3	The agent detects a signal and decides whether it is a tone burst or an impact (see Section 4.2). If an impact, it proceeds to locate and diagnose the impact.	The robot switches its transducer electronics to receive mode, and waits to receive a packet of data.
4	Agent performs a triangulation procedure to determine the position of the robot transducer relative to the four PVDF sensors, based on the differences in the times of arrival of the tone burst.	
5	The agent transmits (using its central PZT transducer) a packet of data that contains: • The coordinates of the robot transducer relative to the centre of the cell. • The gradient field values of the cell and of its (connected) neighbours.	
6	The agent waits for an acknowledgement signal.	The robot receives the data packet. If a packet is not received, the robot makes a small random move of its foot and tries Step 2 again.
7		The robot sends an acknowledge signal and, based on the data in the packet, decides on its next action. Alternatives are as follows: • Reposition its foot on the cell if it is sufficiently far off-centre that it cannot make the next step. • Step to the next cell as indicated by what it determines to be the highest-priority gradient field values. • Cease stepping and manipulate the camera (or other sensors).
8	If acknowledgement received, go back to Step 1. If not, wait for a preset time (∼0.5 s), then retransmit packet as in Step 5. If acknowledgement still not received, go back to Step 1.	

to which the relevant robot foot is attached. When the cell is transmitting, the carrier is generated by the cell's centre PZT transducer and received by the robot's transducer.

The carrier is Binary Phase Shift Key (BPSK)-modulated at a rate of 100 baud, which is slow enough to allow ultrasonic reflections within the panel to decay before the next symbol is sent. The effective data rate for the channel is approximately 80 bits/s, the reduced rate compared to the baud rate being due mainly to the synchronization and checksum bits used.

4.3.4 Motion: Stepping and Navigation

The robot can move with six degrees of freedom: it can rotate each leg about a horizontal axis, and each foot has two independent rotations, one about a horizontal axis and one about a vertical axis. The higher-level steps are defined in terms of these fundamental motions.

Because the robot has no global navigation capabilities and can only move from one cell to the next using dead reckoning, large positional errors could rapidly accumulate as the robot moves over the surface. To avoid such positional errors, the underlying cell measures the robot's foot position by triangulation, as described above, and reports it to the robot. The robot can then either physically correct its foot position or take it into account in calculating the step required for the next move.

A complication with this method of positional feedback is that the cell and the robot must have common knowledge of the orientation of the cell relative to other cells and the structure. One way to avoid this issue is to build the CD structure with all cells in a prescribed orientation. However, it can be argued that this solution is inconsistent with the concept of an adaptive, self-organizing structure, and a more satisfactory solution involves the cells cooperatively determining their relative orientations. This is also necessary for algorithms such as ant colony optimization.

Nevertheless, there is a need for the robot to have some basic knowledge about the structure, since it cannot be allowed to step on the gaps between panels, and it needs to know where the face edges of the prism are located in order to be able to step from one face to another. A robot with more computational power than the present one could, for example, use its video camera to resolve local issues such as these, but for the time being the robot has been given this basic knowledge of the cell layouts.

The robot's navigation and functions are determined cooperatively with the local agents embedded in the test-bed skin with which it is in contact. The robot navigates around the surface of the test bed using gradient field data available from the underlying cell to which it is attached at the time. These data are specific to the cell's local neighbourhood and do not contain any global information about the system. Further information about the gradient fields was provided in Section 4.2 above.

4.3.5 Manipulation of Camera and Image Data Transmission

It is intended that the robot will use its video cameras to image the "damage" (at this stage a red or green spot stuck to the skin at the impact location), but the implementation of this has not yet been completed. The concept is that when the robot arrives at

the neighbourhood of a damaged cell, as indicated by a minimum (zero) in the gradient field, instead of putting its foot down on the damaged cell it will hold it above the cell in an orientation such that the video camera (Fig. 4.7) can be directed at the damage. A routine has been written to recognize the colour of the damage indicator spot from its video image.

In the initial implementation of this damage-imaging capability the data will be sent via a 2.4-GHz wireless link to the system visualizer PC for analysis, but the longer-term aim is for the image data analysis to be carried out on the robot. Use of the visualizer for this function would give it an essential role in the operation of the system, which is not intended. It also creates a potential single point of failure. While at present the robot represents a single point of system failure, the intention is to eventually have large numbers of robots that can share the damage evaluation task and minimize the system's vulnerability to robot failure.

4.4 The System Visualizer

As described in earlier reports (Price et al. 2004; Prokopenko et al. 2006), the system visualizer is a computer that is used for initializing the multiagent system and displaying the state of the system, but plays no essential role in the operation of the system. It can be attached via a USB port at a number of points around the edge of the CD system (in principle it could be anywhere), and its function is to request state information from the embedded agents and display it.

The visualization function has now been extended to show the robot position, and this is illustrated in Fig. 4.10. This information is not obtained from the robot itself,

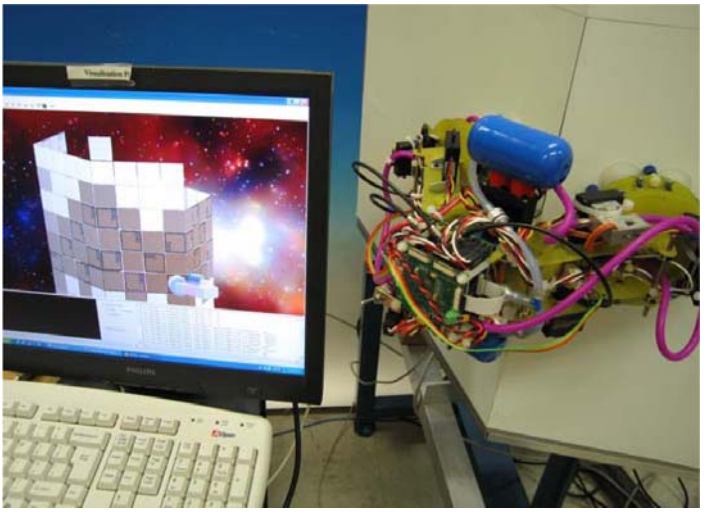


Fig. 4.10. Visualizer screen (left), showing an animation of the robot in its actual position on the CD structure (right).

which in principle does not need to know its absolute position on the structure, but from the agents with which the robot is communicating. Thus, an observer does not need to be able to see the robot to be able to monitor its activities. This principle can of course be extended to more than one robot.

Four views of the concept demonstrator have been developed to display and debug the gradient algorithm. Each view uses a different colour and represents the value of the gradient field on a particular cell by the shade of colour (Fig. 4.5). Lighter shades represent higher gradient values, which are further away from impact locations. Cells that the robot cannot reach because they have no DAS board attached are displayed in white. By double-clicking on a cell in the display the numerical values of the cell's gradients are displayed.

As was pointed out in the previous section, it is intended that, as an initial short-term measure, the visualizer will analyze video image data from the robot to determine the severity of the damage. This is to be used to confirm (or otherwise) the damage diagnosis made by utilizing impact data from the embedded piezoelectric sensors. The longer-term intention is for this analysis to be carried out by the robot (or robots).

This image analysis role is enabled by an image capture card with a 2.4-GHz wireless link to the video cameras on each foot of the robot. The images transmitted from the robot can be viewed in a new window that has been incorporated into the visualizer display. A Matlab routine that enables the colour of the "damage" indicator (as described in Sec. 4.2) to be identified has been compiled and added to the visualizer.

In the present simplified situation, identification of the colour of the damage indicator will provide an authoritative diagnosis of the damage that will supersede the tentative diagnosis made from data from the embedded piezoelectric sensors. However, the situation may not always be as clear-cut as this. An optical image may underestimate internal structural damage within a material (e.g., in laminated composites), and the use of other sensor modalities for this secondary sensing may also give rise to ambiguities or uncertainties in some cases. In the longer term, it will be desirable to make a diagnostic decision based on all of the available sensor data, possibly with the assistance of material or structural models. The use of data from multiple sensing modalities for damage diagnosis and prognosis will be the subject of future work.

In cases when major damage occurs suddenly, which will usually be the result of an external influence such as an impact, it may be necessary to initiate a response to an initial indicator (such as the sensing of the impact) without waiting for a subsequent detailed inspection and diagnosis. In such a perceived emergency situation, a precautionary principle must clearly be adopted; rapid action should be taken first and a more detailed diagnosis made later.

4.5 Conclusions

This chapter has described the development of the first stages of an experimental structural health monitoring system whose operation is based on self-organization in a complex multiagent system. Damage identification, location, and the first stage

of evaluation have been demonstrated, as has the deployment of a secondary robotic inspector. This is all achieved without central control.

The main thrust of recent work has been the development of a mobile robotic agent and the hardware and software modifications and developments required to enable the fixed and mobile agents to operate as a single, self-organizing, multiagent system. This single-robot system is seen as the forerunner of one in which larger numbers of small robots perform inspection and repair tasks cooperatively, by self-organization.

While the goal of demonstrating self-organized damage diagnosis has not yet been fully achieved, much of the work required for the final element—enabling the robot to point the video camera and transmit an image—has been either completed or planned. It is expected that it will be completed shortly. Nevertheless, what has already been achieved is an important step.

The next steps in the development of the system are as follows:

- Completion of the present task of achieving camera operation and integration of diagnostic information.
- Adaptation of the system to incorporate a more realistic damage scenario. Work on corrosion monitoring at 'hot spots' in aircraft is in progress (Muster et al. 2005; Trego et al. 2005), and health monitoring of thermal protection systems in vehicles such as the space shuttle is being undertaken.
- Damage diagnosis based on data from multiple sensing modalities. This will be initially addressed through work on a more realistic damage scenario, but the development of a general framework for the use of multiple data sets for damage diagnosis is a significant problem.
- Expansion of the single-robot capability to accommodate multiple robots that are capable of cooperative solution of tasks such as inspection and repairs.
- While these are all substantial steps, the developments reported here represent an important advance and a sound base from which to make further progress.

While the present demonstration system is clearly not suitable for large-scale implementation in a current aerospace vehicle, it is envisaged that the sensing, computation, and self-repair functions of the embedded system will eventually be integrated into advanced materials. Recent advances in materials science and nanotechnology give confidence that this will be achieved in the foreseeable future, as will the development of microsized, intelligent robots. We believe that the basic approach outlined in this chapter—developing self-organizing, adaptive solutions in distributed multiagent systems—will form the basis of future developments in this area.

Conversely, it is our view that structural health monitoring is an interesting and fertile application area in which to study engineered self-organization. The wide range of spatial and temporal scales on which events can occur and damage develop, and the consequent variety of responses and response requirements, ensure that this general application will provide a more complete challenge for self-organized sensing and response than many others.

Acknowledgements

It is a pleasure to acknowledge the continued support for this work of Drs. Ed Generazio and Bill Prosser of NASA Langley Research Center. We also gratefully acknowledge the contributions of Adam Batten, Graeme Edwards, Tony Farmer, Peter Isaacs and Richard Moore (all from CSIRO Industrial Physics), and Mikhail Prokopenko (CSIRO ICT Centre) to this work.

References

Camazine, S., Deneubourg, J-L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E. (2001). *Self-Organization in Biological Systems.* Princeton University Press, Princeton, NJ.

Dorigo, M., and Di Caro, G. (1999). Ant Colony Optimization: A New Meta-Heuristic, (1999). *Congress on Evolutionary Computation*, pp. 1470–1477, Washington D.C., July 1999.

Dorigo, M., and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge, MA.

Hedley, M., Johnson, M.E., Lewis, C.J., Carpenter, D.A., Lovatt, H., Price, D.C. (2003). *Smart Sensor Network for Space Vehicle Monitoring, Proceedings of the International Signal Processing Conference*. Dallas, TX, March 2003. http://www.gspx.com/GSPX/papers_online/papers_list.php

Hoschke, N., Lewis, C.J., Price, D.C., Scott, D.A., Edwards, G.C., Batten, A. (2006). A self-organising sensing system for structural health management. In Gabrys, B., Howlett, R.J., and Jain, L.C., editors, *Knowledge-Based Intelligent Information and Engineering Systems, 10th International Conference, KES 2006, Bournemouth, UK, October 9-11 2006, Proceedings, Part III*, volume 4253 of *Lecture Notes in Artificial Intelligence*, pages 349–357. Springer, Berlin.

Kohonen, T. (2001). *Self-organizing Maps*. Springer, Heidelberg (3rd Ed.).

Kohonen, T. (2003). Self-organized maps of sensory events, *Philosophical Transactions of the Royal Society, Lond. A*, 361:1177–1186.

Muster, T., Cole, I., Ganther, W., Paterson, D., Corrigan, P. and Price, D. (2005). Establishing a physical basis for the in-situ monitoring of airframe corrosion using intelligent sensor networks, *Proceedings of the 2005 Tri-Service Corrosion Conference*. Orlando, FL, November 2005.

Price, D.C., Batten, A., Edwards, G.C., Farmer, A.J.D., Gerasimov, V., Hedley, M., Hoschke, N., Johnson, M.E., Lewis, C.J., Murdoch, A., Prokopenko, M., Scott, D.A., Valencia, P. and Wang, P. (2004). Detection, evaluation and diagnosis of impact damage in a complex multi-agent structural health management system. In *Proceedings of the 2nd Australasian Workshop on Structural Health Monitoring, Melbourne, December 2004*, pp. 16–27.

Prokopenko, M., Boschetti, F., Ryan, A.J. (2007). An information-theoretic primer on complexity, self-organisation and emergence. Submitted to *Advances in Complex Systems*.

Prokopenko, M., Poulton, G., Price, D.C., Wang, P., Valencia, P., Hoschke, N., Farmer, A.J.D., Hedley, M., Lewis, C., Scott, D.A. (2006). Self-organising impact sensing networks in robust aerospace vehicles. In Fulcher, J., editor, *Advances in Applied Artificial Intelligence*, pages 186–233. Idea Group, Hershey, PA.

Prokopenko, M., Wang, P., Foreman, M., Valencia, P., Price, D. and Poulton, G. (2005a). On connectivity of reconfigurable impact networks in ageless aerospace vehicles. *Journal of Robotics and Autonomous Systems*, 53(1):36–58.

Prokopenko, M., Wang, P., Scott, D.A., Gerasimov, V., Hoschke, N., Price, D.C. (2005b). On self-organising diagnostics in impact sensing networks. In Khosla, R., Howlett, R. J., and Jain, L. C., eds., *Knowledge-Based Intelligent Information and Engineering Systems, 9th International Conference, KES 2005, Melbourne, Australia, September 14-16, 2005, Proceedings, Part IV*, volume 3684 of Lecture Notes in Computer Science, pages 170–178. Springer, Heidelberg.

Rose, J. L. (1999). *Ultrasonic Waves in Solid Media*. Cambridge University Press, Cambridge, UK.

Scott, D.A., Batten, A., Edwards, G.C., Farmer, A.J., Hedley, M., Hoschke, N., Isaacs, P., Johnson, M., Murdoch, A., Lewis, C., Price, D.C., Prokopenko, M., Valencia, P., Wang, P. (2005). An intelligent sensor system for detection and evaluation of particle impact damage, *Review of Progress in Quantitative Nondestructive Evaluation*, 24:1825–32. In Thompson, D.O. and Chimenti, D.E., editors, *American Institute of Physics Conference Proceedings*, volume 760.

Trego, A., Price, D., Hedley, M., Corrigan, P., Cole, I. and Muster, T. (2005). Development of a system for corrosion diagnostics and prognostics, *Proceedings of 1st World Congress on Corrosion in the Military: Cost Reduction Strategies, Sorrento, Italy, June 2005*.

5

Decentralized Decision Making for Multiagent Systems

George Mathews, Hugh Durrant-Whyte, and Mikhail Prokopenko

5.1 Introduction

Decision making in large distributed multiagent systems is a difficult problem. In general, for an agent to make a good decision, it must consider the decisions of all the other agents in the system. This coupling among decision makers has two main causes: (i) the agents share a common objective function (e.g., in a team), or (ii) the agents share constraints (e.g., they must cooperate in sharing a finite resource).

The classical approach to this type of problem is to collect all the information from the agents in a single center and solve the resulting optimization problem [see, e.g., Furukawa et al. (2003)]. However, this centralized approach has two main difficulties:

- The required communication bandwidth grows at least linearly with the number of agents.
- The resulting optimization complexity is generally exponential in the number of agents.

Thus, for a sufficiently large number of agents this problem becomes impractical to solve in a centralized fashion.

However, these difficulties can be overcome by allowing the agents to cooperate or self-organize in solving this distributed decision problem. The main issue in this decentralized approach is identifying which agents need to communicate, what information should be sent, and how frequently.

This chapter approaches the multiagent collaboration problem using analytical techniques and requires that several assumption be made about the form of the problem: (i) the decisions of the individual agents are represented by elements of a continuous and finite-dimensional vector space; (ii) the agents are coupled via a shared objective function that is continuous and twice differentiable, and (iii) there are no interagent constraints.

With these assumptions, this chapter presents fundamental results on the structure of the decentralized optimal decision problem. A simple and intuitive decentralized negotiation algorithm is presented which enables multiple decision makers to propose

and refine decisions to optimize a given team objective function. A convergence analysis of this procedure provides an intuitive relationship between the communication frequency, transmission delays, and the inherent interagent coupling in the system.

The algorithm is applied to the control of multiple mobile robots undertaking an information-gathering task. The specific scenario considered requires that the robots actively localize a group of objects. For this scenario the interagent coupling loosely relates to the amount of overlap between the information that two agents receive when undertaking their respective plans. This requires communications only between coupled agents and results in a more scalable system.

Section 5.2 defines the multiagent decision problem and introduces the decentralized optimization algorithm to solve it. This section also defines the interagent coupling metric and its relationship to the communication structure. Section 5.3 examines the full dependency between the rate an agent can refine its decision with the communication frequency, transmission delays, and the interagent coupling of the system. A decomposition of the objective function is introduced in Section 5.4 that is used to explicitly specify what each agent must communicate. An approximation technique is proposed to enable each agent to calculate the interagent coupling on-line. An overview of the decentralized decision-making or negotiation algorithm is given in Section 5.5. This summarizes exactly what each agent must know about the other agents and details the local procedure executed by each agent. Section 5.6 describes the general multiagent information-gathering control problem and formulates it as a decentralized sequential decision problem. This is specialized for an object localization problem in Section 5.7, with results given in Section 5.8. Section 5.9 provides a summary and directions for future work.

5.2 Asynchronous Decision Making

Consider a system of p agents at some specific instant in time; each agent i is in charge of a local decision variable $\mathbf{v}_i \in \mathcal{V}_i$. As stated in the Introduction, it is assumed that the set of feasible decisions for agent i is a Euclidean space of dimension n_i, i.e., $\mathcal{V}_i \equiv \Re^{n_i}$. This may be relaxed such that $\mathcal{V}_i$ is a convex subset of $\Re^{n_i}$, but for simplicity this case is ignored. The global decision vector, $\mathbf{v} = [\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_p^T]^T$, is defined on the product space $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_p = \Re^n$, where $n = n_1 + \cdots + n_p$.

The system as a whole is required to select the decisions such that a given objective function $J : \mathcal{V}_1 \times \cdots \times \mathcal{V}_p \rightarrow \Re$ is minimized. The objective function captures the goals of the system and will generally incorporate a model of how the agents interact with the environment using their sensors and actuators. The optimal decision problem is given by

$$\mathbf{v}^* = \arg\min_{\mathbf{v} \in \mathcal{V}} J(\mathbf{v}), \tag{5.1}$$

where $\mathbf{v}^*$ is the desired optimal global decision.

Assumption 1 (Objective Function) *The objective function J is twice differentiable, convex, and bounded from below.*

Under the convexity assumption, Eq. (5.1) is equivalent to requiring the gradient vector to vanish:

$$\nabla J(\mathbf{v}^*) = \mathbf{0}. \tag{5.2}$$

In terms of each agent's local decision, this can be written as

$$\nabla_i J(\mathbf{v}^*) = \mathbf{0} \quad \forall\, i, \tag{5.3}$$

where $\nabla_i J(\mathbf{v}) \in \Re^{n_i}$ represents the components of the gradient vector in the subspace $\mathcal{V}_i$. It is this optimality condition that is considered throughout this chapter.

5.2.1 Local Decision Refinement

The proposed solution method for the optimization problem allows each agent to submit an initial decision and then to incrementally refine this, while intermittently communicating these refinements to the rest of the system. The distributed nature of the problem requires that each agent execute and communicate asynchronously; thus the information it has about other agents may be outdated. This requires that each agent maintain a local copy of the team decision vector, which is given at a discrete time t for each agent i as

$$\begin{aligned} {}^i\mathbf{v}(t) &= \left[{}^i\mathbf{v}_1(t), \ldots, {}^i\mathbf{v}_p(t)\right] && (5.4)\\ &= \left[{}^1\mathbf{v}_1\big(\tau_{1i}(t)\big), \ldots, {}^p\mathbf{v}_p\big(\tau_{pi}(t)\big)\right]. && (5.5) \end{aligned}$$

In general a presuperscript represents a copy held by a specific agent, while a subscript represents a specific agent's decision [e.g., ${}^i\mathbf{v}_j(t)$ represents agent i's local copy of agent j's decision]. The variable $\tau_{ji}(t)$ in Eq. (5.5) represents the time agent i's local copy ${}^i\mathbf{v}_j(t)$ was generated by agent j and hence ${}^i\mathbf{v}_j(t) = {}^j\mathbf{v}_j\big(\tau_{ji}(t)\big)$. It is assumed that $\tau_{ii}(t) = t$, and thus agent i always has the latest copy of its decision variable.

The time variable t is used simply to represent when discrete events take place (such as when an agent computes an update or communicates) and does not require each agent to have access to a global clock or to perform a periodic synchronization.

To formalize the intuitive notion of *decision refinement*, a local update rule $f_i \colon \mathcal{V} \to \mathcal{V}_i$ will be defined for each agent that modifies its local decision ${}^i\mathbf{v}_i$, based on its copy of the global decision vector ${}^i\mathbf{v}$. To allow each agent to perform updates asynchronously a set of times T_U^i is associated with each agent i that represents when the agent computes a local update:

$${}^i\mathbf{v}_i(t+1) = \begin{cases} f_i\big({}^i\mathbf{v}(t)\big) & \text{if } t \in T_U^i \\ {}^i\mathbf{v}_i(t) & \text{else} \end{cases}. \tag{5.6}$$

For the update to be beneficial, it should decrease the value of the objective function. Thus, a steepest descent update is used:

$$f_i\big({}^i\mathbf{v}(t)\big) = {}^i\mathbf{v}_i(t) - \gamma_i\, \nabla_i J\big({}^i\mathbf{v}(t)\big), \tag{5.7}$$

where γ_i is a step size.

However, since only local and possibly out-of-date information is available, it is not trivial to guarantee that this will decrease the value of the objective function corresponding to the latest decisions from all agents. The rest of this section develops conditions that will guarantee this property and the overall convergence of the algorithm by providing limits on the step size γ_i.

5.2.2 Communication

Communication is initiated by agent i sending a message, at some time $t \in T_C^{ij}$, to agent j containing its latest decision ${}^i\mathbf{v}_i(t)$. After some communication delay $b_{ij}(t)$, agent j receives it and incorporates it into its local copy of the team decision vector. Thus, when the message is received ${}^j\mathbf{v}_i\big(t + b_{ij}(t)\big) = {}^i\mathbf{v}_i(t)$, and hence $\tau_{ij}\big(t + b_{ij}(t)\big) = t$. For each agent to obtain the decisions of every other agent, each agent must communicate with every other agent.

Assumption 2 (Bounded Delays) *There exist finite positive constants B_{ij} such that*

$$t - \tau_{ij}(t) \leq B_{ij} \quad \forall\, i, j, t. \tag{5.8}$$

Thus, the age of agent j's local copy of agent i's decision is bounded.

Informally, Assumption 2 can be relaxed such that B_{ij} represents the time difference, measured in numbers of updates computed by agent i, between ${}^i\mathbf{v}_i(t)$ and ${}^j\mathbf{v}_i(t)$.

If the agents compute updates and communicate at a fixed frequency, these can be approximated by knowing: (i) the number of iterations R_i computed by agent i per second, (ii) the number of communicated messages $C_{i\to j}$ from i to j per second, and (iii) the delivery delay $D_{i\to j}$ between agent i sending and j receiving a message in seconds, using

$$\hat{B}_{ij} = \frac{R_i}{C_{i\to j}} + R_i D_{i\to j}. \tag{5.9}$$

5.2.3 Interagent Coupling

For each pair of agents, the magnitude of interagent coupling is captured by a single scalar that represents the maximum curvature of the objective function in the subspace containing the decision variables of the two agents.

Assumption 3 (Coupling) *For every i and j, there exists a finite positive constant K_{ij}, such that*

$$\mathbf{x}_i^T\, \nabla_{ij}^2 J(\mathbf{v})\, \mathbf{x}_j \leq \|\mathbf{x}_i\|\, K_{ij}\, \|\mathbf{x}_j\| \tag{5.10}$$

for all $\mathbf{v} \in \mathcal{V}$ and all column vectors $\mathbf{x}_i \in \mathcal{V}_i$ and $\mathbf{x}_j \in \mathcal{V}_j$. The matrix $\nabla_{ij}^2 J(\mathbf{v})$ corresponds to the $n_i \times n_j$ submatrix of the Hessian of the objective function. The vector norms are defined as the Euclidean l^2 norm.

It is noted that the coupling constants are symmetric, and hence $K_{ij} = K_{ji}$.

If the actual Hessian is available, then the limit can be computed using

$$K_{ij} = \max_{\mathbf{v}\in\mathcal{V}} \left\| \nabla^2_{ij} J(\mathbf{v}) \right\|_M , \tag{5.11}$$

where $\|\cdot\|_M$ represents the induced matrix norm and is given by

$$\|A\|_M = \max_{\|x\|=1} \|Ax\| , \tag{5.12}$$

where x is a vector of appropriate dimension.

5.2.4 Asynchronous Convergence

The amount of interagent coupling and the magnitude of the communication delays play an important role in the amount each agent may refine its local decision. Intuitively, if the decisions of two agents are highly dependent, they should communicate more often while refining their decisions.

Equation (5.10) defined a metric K_{ij} measuring interagent coupling for a given multiagent decision problem. Similarly Eq. (5.8) encapsulated the effects of asynchronous execution and communication delays between two agents using B_{ij}. These are now used to provide an intuitive limit on the refinement step size γ_i introduced in Eq. (5.7).

Theorem 1 (Convergence). *Assumptions 1 to 3 provide sufficient conditions for the distributed asynchronous optimization algorithm defined by Eq. (5.7) to converge to the global optimum, defined by Eq. (5.3), for all $\gamma_i \in (0, \Gamma_i)$, where*

$$\Gamma_i = \frac{2}{K_{ii} + \sum_{j\neq i} K_{ij}\left(1 + B_{ij} + B_{ji}\right)}. \tag{5.13}$$

The convergence of this type of algorithm was first proved by Tsitsiklis et al. (1986).

This theorem provides a unified way of relating the inherent interagent coupling and communication structure with the speed at which an agent can refine its local decision.

Based on Theorem 1 an algorithm can be developed by defining the step size as

$$\gamma_i = \frac{\beta}{K_{ii} + \sum_{j\neq i} K_{ij}\left(1 + B_{ij} + B_{ji}\right)} \tag{5.14}$$

for some $\beta \in (0, 2)$.

5.3 Communication Structure

For the general problem under consideration, for each agent to receive the decisions of every other agent, there must exist a communication channel among all the agents in the system. Regardless of the implementation, the only relevant features of this

communication network are the interagent communication frequency and transmission delays.

The communication frequency is directly controllable by the agents, whereas the transmission delays are determined by properties of the communication network and/or the nature of the physical communication medium.

For simplicity, this work assumes that the communication network is fully connected and that the transmission delays are fixed for any pair of agents. Although this may seem unrealistic it will be shown later that for most realistic scenarios each agent will only be coupled to agents in a local neighbourhood and hence will only be required to communicate to them. This assumption also allows the (higher-level) problem of network formation and routing to be ignored.

5.3.1 Communication Rate

The rate at which agents communicate has a significant impact on the convergence rate of the optimization process. Although a detailed analysis of the convergence rate is beyond the scope of this work, it is reasonable to assume that it is proportional to the step size γ_i (larger steps will generally allow the agents' decisions to be refined faster).

The step size for a given agent i is determined by Eq. (5.14), where the delay terms can be approximated by $\hat{B}_{ij}$, defined in Eq. (5.9). When these are combined, the step size obeys the relation

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j \neq i} K_{ij} \left(1 + \frac{R_i}{C_{i \to j}} + R_i D_{i \to j} + \frac{R_j}{C_{j \to i}} + R_j D_{j \to i} \right). \tag{5.15}$$

If it is assumed that the computation rates (R_i and R_j) and interagent communication delays ($D_{j \to i}$ and $D_{i \to j}$) are fixed, this becomes

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j \neq i} K_{ij} \left(W_{ij} + \frac{R_i}{C_{i \to j}} + \frac{R_j}{C_{j \to i}} \right), \tag{5.16}$$

where $W_{ij} = 1 + R_i D_{i \to j} + R_j D_{j \to i}$ and is a constant. Thus, from Eq. (5.16), the step size γ_i is maximized when all the communication rates are also maximized; i.e. every agent communicates to every other agent after every local iteration.

However, this policy is impractical for large systems containing many agents. Potentially this can be overcome by allowing each pair of agents to communicate at a rate proportional to the coupling, i.e.,

$$C_{i \to j} = \eta_i K_{ij} \tag{5.17}$$

for some constant η_i. However, this will also be impractical for large systems since the step size will be directly related to the number of agents. This can be demonstrated by considering all agents to have a fixed computation rate (i.e., $R_i = R$ and $\eta_i = \eta, \forall\, i$) and substituting Eq. (5.17) into Eq. (5.16):

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j \neq i} K_{ij} W_{ij} + 2(p-1)\frac{R}{\eta}. \tag{5.18}$$

Thus, for large p, the last term will dominate causing the step size γ_i to approach 0 regardless of how small the actual interagent coupling may be. Hence, a communication rate proportional to the coupling is generally too low.

To overcome this it is proposed to set the communication rate proportional to the *square root* of the coupling:

$$C_{i\to j} = \eta_i \sqrt{K_{ij}}, \tag{5.19}$$

which represents a compromise between fast convergence, requiring a high communication rate, and the practical requirements of a slow communication rate. The step size γ_i, and hence the possible convergence rate, is now only dependent on the coupling, which is in turn determined by the inherent complexity of the problem. This can be demonstrated by substituting Eq. (5.19) into Eq. (5.16):

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j\neq i}\Big(K_{ij}W_{ij} + \sqrt{K_{ij}}\big(\eta_i R_i + \eta_j R_j\big)\Big). \tag{5.20}$$

The constant η_i can be chosen such that the strongest coupled agent is sent a message after every local iteration or to satisfy some bandwidth limitations.

5.4 Modularization

Until now it has been implicitly assumed that each agent has a full representation of the system's objective function. In general this requires the state and sensor/actuator models of all agents. This may not be a problem for a small number of agents, but poses a significant issue for a large distributed system.

Ideally, each agent should only require a model of itself and relatively little knowledge of the other agents. This issue has been examined by Mathews et al. (2006), who identified a composable or partially separable form of the objective function that enables this type of modularization.

5.4.1 Partial Separability

The partially separable objective function allows the effects or impacts of each agent to be separated and evaluated independently. The objective function then becomes a function of the composition of these individual impacts.

Definition 1 (Partial Separability). *A partially separable*[1] *system has an objective function of the form*

$$J(\mathbf{v}) = \psi\big(\Upsilon_1(\mathbf{v}_1) * \Upsilon_2(\mathbf{v}_2) * \cdots * \Upsilon_p(\mathbf{v}_p)\big), \tag{5.21}$$

where $\Upsilon_i : \mathcal{V}_i \to \mathcal{I}$ *is the* i^{th} *agent's* impact function *and maps a decision to an element of an* impact space $\mathcal{I}$. *An element* $\alpha \in \mathcal{I}$ *of this space will be referred to as an* impact. *The* composition operator $* : \mathcal{I} \times \mathcal{I} \to \mathcal{I}$ *allows impacts to be summarized without losing any task-relevant information. The* generalized objective function $\psi : \mathcal{I} \to \Re$ *maps a combined team impact to a scalar cost.*

[1] The function becomes fully separable if $\mathcal{I} \equiv \Re$, $* \equiv +$ and ψ is linear.

An agent's impact function is an abstraction of its state and sensor/actuator models and maps a decision onto a task-specific impact space. It is assumed that each agent i only has knowledge of its own impact function Υ_i and thus requires the impacts $\alpha_j = \Upsilon_j(\mathbf{v}_j)$ from every other agent $j \neq i$ for the objective function to be evaluated. Thus, instead of each agent maintaining a local copy of every other agent's decision vector $\mathbf{v}_j$, it simply maintains their impact α_j.

This definition allows one to abstract away state and sensor/actuator models of the other agents and defines a common language of impacts that the agents use to communicate. For simplicity, the cumulative composition operator $\bigodot$ will be used, such that Eq. (5.21) can be written as

$$J(\mathbf{v}) = \psi\Big(\bigodot_{i=1}^{p} \Upsilon_i(\mathbf{v}_i)\Big). \tag{5.22}$$

Example: Collision Avoidance

To provide an example of an impact, consider a multiagent path-planning scenario with a separation requirement. For this task the objective function will be dependent on the separation between the agents, which in turn requires the individual paths (possibly represented by a fixed number of points) from all the agents. In this case each agent abstracts away its motion model and corresponding control inputs. Thus, an agent's path can be considered as its impact, and the composition operator simply collects the paths. The generalized objective function is used to calculate the associated cost of these specific paths.

It is noted that for this example the size of the impact space is proportional to the number of agents (the number of paths is the same as the number of agents). This differs from the example presented in Section 5.7, which has a composition operator given by addition. For this case the size of the impact space is independent of the number of agents (the sum of many numbers is still a number).

5.4.2 Modular Decision Refinement

With the use of the partially separable form of the objective function, the local decision refinement process, presented in Section 5.2.1, can be modified such that each agent i is only required to maintain a local copy of the impact of every other agent $\{{}^i\alpha_j(t) : \forall\, j \neq i\}$. This information is updated via messages from the other agents in the system and may contain delayed information (according to Assumption 2).

From this, the local decision refinement equations for agent i [corresponding to Eq. (5.7)] becomes

$$f_i\big({}^i\mathbf{v}(t)\big) = {}^i\mathbf{v}_i(t) - \gamma_i\, \mathbf{s}_i(t), \tag{5.23}$$

where $\mathbf{s}_i(t)$ is the local steepest descent direction $\nabla_i J\big({}^i\mathbf{v}(t)\big)$ and is given by

$$\mathbf{s}_i(t) = \nabla_i \psi\Big(\Upsilon_i\big({}^i\mathbf{v}_i(t)\big) * \bigodot_{j\neq i} {}^i\alpha_j(t)\Big). \tag{5.24}$$

Calculation of the step size γ_i requires the coupling terms K_{ij}. If the system is static, e.g, for a monitoring and control system for a distributed processing plant, these can be calculated in advance during the design phase by solving Eq. (5.11). However, for the majority of multiagent systems this will not be the case and the coupling terms must be calculated on-line.

5.4.3 On-line Coupling Estimation

Since no single agent has an explicit representation of the full objective function, it is not possible for any agent to solve Eq. (5.11) for the coupling terms. Rather, these terms are approximated using a finite difference evaluated locally by each agent from two successive iterations and communications.

Consider the Taylor expansion of J about a decision vector $\mathbf{v}$ with a perturbation $\Delta\mathbf{v} = [\Delta\mathbf{v}_1^T, \ldots, \Delta\mathbf{v}_p^T]^T$:

$$J(\mathbf{v}+\Delta\mathbf{v}) \approx J(\mathbf{v}) + \sum_{i=1}^{p} \Delta\mathbf{v}_i^T \, \nabla_i J(\mathbf{v}) + \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p} \Delta\mathbf{v}_i^T \, \nabla_{ij}^2 J(\mathbf{v}) \, \Delta\mathbf{v}_j. \quad (5.25)$$

The use of the coupling K_{ij} gives a maximum bound on the value of the last term [see Eq. (5.10)]. It is proposed to approximate the coupling by simply estimating this term over successive iterations.

If only perturbations in the decisions of agents i and j are considered, then the cross-derivative term can be estimated using a backwards difference approximation,

$$\Delta\mathbf{v}_i^T \nabla_{ij}^2 J(\mathbf{v}) \Delta\mathbf{v}_j \approx J(\mathbf{v}^{aa}) + J(\mathbf{v}^{bb}) - J(\mathbf{v}^{ba}) - J(\mathbf{v}^{ab}),$$

where $\mathbf{v}^{aa} = \mathbf{v}$, $\mathbf{v}^{ba} = \mathbf{v} - \Delta\mathbf{v}_i$, $\mathbf{v}^{ab} = \mathbf{v} - \Delta\mathbf{v}_j$, and $\mathbf{v}^{bb} = \mathbf{v} - \Delta\mathbf{v}_i - \Delta\mathbf{v}_j$; note that the increments are assumed to be added on to the correct components in the team decision vector.

For a system with a partially separable objective function, the i^{th} agent, at iteration t, can estimate its coupling to any other agent j using the local decisions ${}^i\mathbf{v}_i^b$ and ${}^i\mathbf{v}_i^a$ from the two previous iterations, with corresponding impacts ${}^i\alpha_i^b = \Upsilon_i({}^i\mathbf{v}_i^b)$ and ${}^i\alpha_i^a = \Upsilon_i({}^i\mathbf{v}_i^a)$, and a decision increment with norm $d_i = \|{}^i\mathbf{v}_i^a - {}^i\mathbf{v}_i^b\|$. Also required are the previous two impacts ${}^i\alpha_j^b$ and ${}^i\alpha_j^a$ communicated from agent j with a corresponding decision increment with norm d_j using

$${}^i\hat{K}_{ij}(t) = \left| \frac{\psi({}^i\alpha_T^{aa}) + \psi({}^i\alpha_T^{bb}) - \psi({}^i\alpha_T^{ba}) - \psi({}^i\alpha_T^{ab})}{d_i d_j} \right|, \quad (5.26)$$

where

$${}^i\alpha_T^{aa} = {}^i\alpha_{\bar{i}\bar{j}} * {}^i\alpha_i^a * {}^i\alpha_j^a, \qquad {}^i\alpha_T^{bb} = {}^i\alpha_{\bar{i}\bar{j}} * {}^i\alpha_i^b * {}^i\alpha_j^b,$$
$${}^i\alpha_T^{ba} = {}^i\alpha_{\bar{i}\bar{j}} * {}^i\alpha_i^b * {}^i\alpha_j^a, \qquad {}^i\alpha_T^{ab} = {}^i\alpha_{\bar{i}\bar{j}} * {}^i\alpha_i^a * {}^i\alpha_j^b$$

and ${}^i\alpha_{\bar{i}\bar{j}}$ is the combined impact of the rest of the team:

$${}^i\alpha_{\bar{i}\bar{j}} = \bigodot_{q \neq i,j} {}^i\alpha_q^a.$$

This approximation requires that each agent communicate its local impact ${}^i\alpha_i \in \mathfrak{I}$ and the amount its decision has changed since the last communication $d_i \in \Re^+$. It is also noted that the coupling can only be approximated after two messages have been received.

This method allows the coupling estimates ${}^i\hat{K}_{ij}(t)$ to track the curvature of the objective function in the vicinity of the actual team decision vector and in the directions in which the decisions are actively being refined. This is in contrast to using an absolute maximum over all positions and directions, as suggested in Eq. (5.10).

This approximation is used to calculate the step size γ_i of the subsequent iteration:

$$\gamma_i(t) = \frac{\beta}{{}^i\hat{K}_{ii}(t) + \sum_{j \neq i} {}^i\hat{K}_{ij}(t)\big(1 + \hat{B}_{ij} + \hat{B}_{ji}\big)}. \tag{5.27}$$

The term ${}^i\hat{K}_{ii}(t)$, representing the maximum curvature in the subspace of agent i's decision, is approximated directly by agent i with

$${}^i\hat{K}_{ii}(t) = \Big\| \nabla^2_{ii}\psi\big(\Upsilon_i({}^i\mathbf{v}_i^a) * \bigodot_{j \neq i} {}^i\alpha_j^a\big) \Big\|_M.$$

Or, if the Hessian submatrix cannot be calculated directly, ${}^i\hat{K}_{ii}$ can be calculated using a finite difference approach in a manner similar that for ${}^i\hat{K}_{ij}$.

The value of β can be large for problems with slowly varying Hessians, small communication delays, and high communication rates. However, for large delays or rapidly varying Hessians, a value closer to zero should be employed. For the results presented in this chapter a value of $\beta = 1$ was used.

5.4.4 Dynamic Communication

Equation (5.26) allows each agent to approximate the interagent coupling at each iteration t of the algorithm. This, in turn, can be used to calculate an appropriate communication rate using Eq. (5.19). Although this violates the assumption of a fixed communication rate, which is made when approximating B_{ij} in Eq. (5.9), it will be approximately true for the iterations around t.

This allows the interagent communication rate to dynamically vary throughout the operation of the algorithm in response to changing interagent coupling.

5.5 Algorithmic Details

A brief illustration of the structure of the final decentralized algorithm is given in Fig. 5.1. Table 5.1 lists all the functions and variables an agent must maintain, while the full details of the local algorithm executed by each agent is given by Algorithm 5.1.

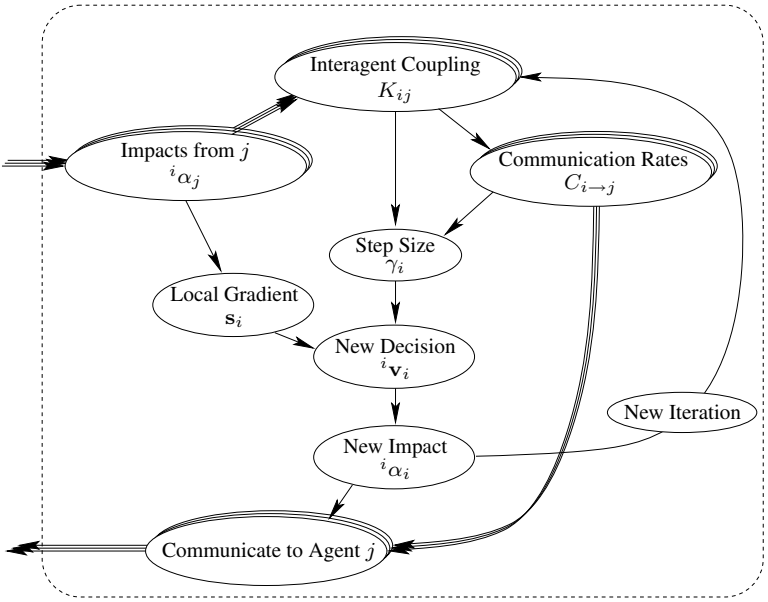


Fig. 5.1. General structure of the local algorithm executed by the i^{th} agent. For full details see Algorithm 5.1.

Table 5.1. Functions and variables maintained by the i^{th} agent.

Functions	Description
$\Upsilon_i \colon \mathcal{V}_i \to \mathcal{I}$	Local impact function (abstraction of sensor and actuator models)
$* \colon \mathcal{I} \times \mathcal{I} \to \mathcal{I}$	Impact composition operator
$\psi : \mathcal{I} \to \Re$	Generalized objective function
Variables	**Description**
${}^i\mathbf{v}_i^a \in \mathcal{V}_i$	Local decision
$d_i \in \Re^+$	Distance local decision moved during last iteration
$\{{}^i\mathbf{v}_i^{\to j} \in \mathcal{V}_i : \forall\, j \neq i\}$	Set of previous local decisions, corresponding to communication events to agent j
${}^i\alpha_i^a \in \mathcal{I}$	Local impact
${}^i\alpha_i^b \in \mathcal{I}$	Local impact from previous iteration
$\{{}^i\alpha_j^a \in \mathcal{I} : \forall\, j \neq i\}$	Set of impacts from other agents
$\{{}^i\alpha_j^b \in \mathcal{I} : \forall\, j \neq i\}$	Set of previous impacts from other agents
$\{d_j \in \Re^+ : \forall\, j \neq i\}$	Set of distances other agents decisions moved between last two communications
$\{R_j \in \Re^+ : \forall\, j\}$	Set of computation rates
$\{C_{i\to j} \in \Re^+ : \forall\, j \neq i\}$, $\{C_{j\to i} \in \Re^+ : \forall\, j \neq i\}$	Sets of communication rates
$\{D_{i\to j} \in \Re^+ : \forall\, j \neq i\}$, $\{D_{j\to i} \in \Re^+ : \forall\, j \neq i\}$	Sets of transmission delays

Algorithm 5.1: Local algorithm run by the i^{th} agent.

for all $j \neq i$ **do**
  Initialise communication link to j
  Determine communication delays $D_{i\to j}$ and $D_{j\to i}$
  Exchange computation rates R_i and R_j
  $\text{NumMsg}_j \Leftarrow 0$
end for
Initialise decision ${}^i\mathbf{v}_i^a$ *(e.g. using* ${}^i\mathbf{v}_i^a \Leftarrow \arg\min_{\mathbf{v}_i \in \mathbb{V}_i} \psi\big(\Upsilon_i(\mathbf{v}_i)\big)$*)*
repeat
  for all $j \neq i$ **do** *(Estimate interagent coupling)*
    if $\text{NumMsg}_j < 2$ **then**
      ${}^i\hat{K}_{ij} \Leftarrow 0$
      $\hat{B}_{ij} \Leftarrow 0$
      $\hat{B}_{ji} \Leftarrow 0$
    else
      Evaluate ${}^i\hat{K}_{ij}$ using Eq. (5.26)
      $\hat{B}_{ij} \Leftarrow R_i/C_{i\to j} + R_i D_{i\to j}$
      $\hat{B}_{ji} \Leftarrow R_j/C_{j\to i} + R_j D_{j\to i}$
    end if
  end for
  ${}^i\hat{K}_{ii} \Leftarrow \left\| \nabla_{ii}^2 \psi\big(\Upsilon_i({}^i\mathbf{v}_i^a) * \bigodot_{j\neq i} {}^i\alpha_j^a\big) \right\|_M$
  $\gamma_i \Leftarrow \dfrac{\beta}{{}^i\hat{K}_{ii} + \sum_{j\neq i} {}^i\hat{K}_{ij}(1 + \hat{B}_{ij} + \hat{B}_{ji})}$ *(Evaluate step size)*
  $\mathbf{s}_i \Leftarrow -\nabla_i \psi\big(\Upsilon_i({}^i\mathbf{v}_i^a) * \bigodot_{j\neq i} {}^i\alpha_j^a\big)$ *(Evaluate update direction)*
  ${}^i\mathbf{v}_i^a \Leftarrow {}^i\mathbf{v}_i^a + \gamma_i \mathbf{s}_i$ *(Update local decision)*
  $d_i \Leftarrow \|\gamma_i \mathbf{s}_i\|$ *(Save distance decision moved)*
  ${}^i\alpha_i^b \Leftarrow {}^i\alpha_i^a$ *(Save previous local impact)*
  ${}^i\alpha_i^a = \Upsilon_i({}^i\mathbf{v}_i^a)$ *(Evaluate new local impact)*
  for all $j \neq i$ **do** *(Manage communications)*
    $C_{i\to j} \Leftarrow \eta\sqrt{{}^i\hat{K}_{ij}}$ *($\eta = R_i/\max_{j\neq i}\sqrt{{}^i\hat{K}_{ij}}$ or is determined by some bandwidth constraint)*
    if Required to send message to j **then** *(Determined from $C_{i\to j}$)*
      Send $m_{ij} = \{{}^i\alpha_i^a, \|{}^i\mathbf{v}_i^a - {}^i\mathbf{v}_i^{\to j}\|, C_{i\to j}\}$ to j
      ${}^i\mathbf{v}_i^{\to j} \Leftarrow {}^i\mathbf{v}_i^a$ *(Save decision for use in next message)*
    end if
    if Message $m_{ji} = \{{}^j\alpha_j^m, d_j^m, C_{j\to i}^m\}$ received from j **then**
      ${}^i\alpha_j^b \Leftarrow {}^i\alpha_j^a$ *(Save previous impact)*
      ${}^i\alpha_j^a \Leftarrow {}^j\alpha_j^m$ *(Update current impact)*
      $d_j \Leftarrow d_j^m$ *(Update the distance the decision moved)*
      $C_{j\to i} \Leftarrow C_{j\to i}^m$ *(Update the communication rate)*
      $\text{NumMsg}_j \Leftarrow \text{NumMsg}_j + 1$
    end if
  end for
until Converged

5.5.1 Convergence

Since this algorithm is based on an approximation of Theorem 1, it is no longer theoretically guaranteed to converge, but experience has shown that it still converges for a wide range of problems. This is because the theorem only presents a sufficient condition for convergence and is in general overly pessimistic.

5.5.2 Termination

If the algorithm converges, it will reach the optimum set of decision only in the limit of infinite iterations. However, due to the gradient descent property of the algorithm, each iteration will, on average, reduce the system cost. This ensures that the quality of the decisions will continue to improve over time. Thus, for time-critical applications it is reasonable for each agent to simply terminate the algorithm and use the final decision generated.

5.6 Active Information Gathering

The application considered in this work consists of multiple mobile robots undertaking a reconnaissance or information-gathering task. This type of scenario requires the agents to actively gather information on a particular external random variable $\mathbf{x} \in \mathcal{X}$. In general this may include the positions and identities of stationary or mobile objects, terrain properties of a given region, or even surface information of a remote planet. In Section 5.7 this will be specialized for the active localization of a group of objects.

5.6.1 Agents

The mobile robotic agents are modelled as discrete time dynamical systems, with the i^{th} agent's state given by $\mathbf{s}_i \in \mathcal{S}_i$. The agent is controlled from discrete time $k-1$ to k by applying a particular control input $\mathbf{u}_i^k \in \mathcal{U}_i$. In general this causes the agent's state to change according to the probabilistic discrete time Markov motion model $P(\mathbf{s}_i^k|\mathbf{s}_i^{k-1}, \mathbf{u}_i^k)$. However, for simplicity it is assumed that the agent's motion is known with precision, i.e., $\mathbf{s}_i^k = \mathbf{f}_i(\mathbf{s}_i^{k-1}, \mathbf{u}_i^k)$. The joint system state and transition model is given by $\mathbf{s}^k = \mathbf{f}(\mathbf{s}^{k-1}, \mathbf{u}^k) = \{\mathbf{f}_1(\mathbf{s}_1^{k-1}, \mathbf{u}_1^k), \ldots, \mathbf{f}_p(\mathbf{s}_p^{k-1}, \mathbf{u}_p^k)\} = \{\mathbf{s}_1^k, \ldots, \mathbf{s}_p^k\}$.

The observations made by the i^{th} agent are modelled by the conditional density $P(\mathbf{z}_i^k|\mathbf{x}^k; \mathbf{s}_i^k)$, which describes the probability of obtaining a particular observation $\mathbf{z}_i^k$ given the external state $\mathbf{x}^k$ and agents state $\mathbf{s}_i^k$. The notation $\mathbf{z}^k$ denotes the set of observations from all the agents at time step k, i.e., $\mathbf{z}^k = \{\mathbf{z}_1^k, \ldots, \mathbf{z}_p^k\} \in \mathcal{Z} = \prod_{i=1}^p \mathcal{Z}_i$. With the assumption that the observations are conditionally independent given the states $\mathbf{x}^k$ and $\mathbf{s}_i^k$, the combined sensor model can be written as $P(\mathbf{z}^k|\mathbf{x}^k; \mathbf{s}^k) = \prod_{i=1}^p P(\mathbf{z}_i^k|\mathbf{x}^k; \mathbf{s}_i^k)$.

The agents are to be controlled such that the combined observations they receive produce the most informative (or least uncertain) belief about $\mathbf{x}$. To accomplish this

the distribution's entropy will be used as a measure of its associated uncertainty. The entropy of a distribution $P(\mathbf{x})$ is the negative of the expectation of its logarithm:

$$H_{P(\mathbf{x})} = -\mathrm{E}_{\mathbf{x}}[\log P(\mathbf{x})]. \tag{5.28}$$

This can be used for continuous variables, where $P(\mathbf{x})$ is the probability density function, or for discrete variables, where $P(\mathbf{x})$ is the probability distribution function. For a detailed description of the properties of this metric see Cover and Thomas (1991).

5.6.2 Bayesian Filtering

This section details the process of maintaining an accurate belief (as a probability) about the state $\mathbf{x}$. The Bayesian approach requires that the system be given some prior belief; if nothing is known this may simply be a noninformative uniform prior. Once this has been established, the belief at any later stage can be constructed recursively. To avoid potential confusion, instantiated variables (variables assuming a specific value) will be denoted using a tilde, e.g., $P(\tilde{\mathbf{x}}) \equiv P(\mathbf{x} = \tilde{\mathbf{x}})$.

Consider the system at a given time step k. The system's state is given by $\tilde{\mathbf{s}}^k$ and the belief about $\mathbf{x}^k$, conditioned on all past observations and agent configurations, is $P(\mathbf{x}^k|\tilde{\mathbf{Z}}^k; \tilde{\mathbf{S}}^k)$, where $\tilde{\mathbf{Z}}^k = \{\tilde{\mathbf{z}}^1, \ldots, \tilde{\mathbf{z}}^k\}$ and $\tilde{\mathbf{S}}^k = \{\tilde{\mathbf{s}}^1, \ldots, \tilde{\mathbf{s}}^k\}$ and $\tilde{\mathbf{Z}}^0 = \tilde{\mathbf{S}}^0 = \{\emptyset\}$.

When a joint control action, $\tilde{\mathbf{u}}^{k+1}$ is taken, the new state of the agents becomes $\tilde{\mathbf{s}}^{k+1} = \mathbf{f}(\tilde{\mathbf{s}}^k, \tilde{\mathbf{u}}^{k+1})$, and an observation $\tilde{\mathbf{z}}^{k+1}$ is taken. To update the belief about $\mathbf{x}^{k+1}$, it must first be predicted forwards in time using the Chapman-Kolmogorov equation:

$$P(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^k; \tilde{\mathbf{S}}^k) = \int_{\mathcal{X}} P(\mathbf{x}^{k+1}|\mathbf{x}^k) P(\mathbf{x}^k|\tilde{\mathbf{Z}}^k; \tilde{\mathbf{S}}^k) d\mathbf{x}^k. \tag{5.29}$$

The belief can now be updated using Bayes rule:

$$P(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^{k+1}; \tilde{\mathbf{S}}^{k+1}) = \frac{1}{\mu} P(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^k; \tilde{\mathbf{S}}^k) \prod_{i=1}^{p} P(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1}; \tilde{\mathbf{s}}_i^{k+1}), \tag{5.30}$$

where $\tilde{\mathbf{z}}^{k+1} = \{\tilde{\mathbf{z}}_1^{k+1}, \ldots, \tilde{\mathbf{z}}_p^{k+1}\}$ are the actual observations taken by the agents. The term $P(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1}, \tilde{\mathbf{s}}_i^{k+1})$ is the i^{th} agent's observation model evaluated at the actual observation and agent configuration, resulting in a likelihood over $\mathbf{x}^{k+1}$. The normalization constant μ is given by

$$\begin{aligned} \mu &= P(\tilde{\mathbf{z}}^{k+1}|\tilde{\mathbf{Z}}^k; \tilde{\mathbf{S}}^{k+1}) \\ &= \int_{\mathcal{X}} P(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^k; \tilde{\mathbf{S}}^k) \prod_{i=1}^{p} P(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1}; \tilde{\mathbf{s}}_i^{k+1}) d\mathbf{x}^{k+1}. \end{aligned} \tag{5.31}$$

For all agents to maintain this belief, each agent must communicate the observation likelihood function $\lambda(\mathbf{x}^{k+1}) = P(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1}, \tilde{\mathbf{s}}_i^{k+1})$ after each observation is made. The field of decentralized data fusion examines efficient ways for this to be communicated around a sensor network (Manyika and Durrant-Whyte 1994; Liggins et al.

1997); however for this work it is assumed that each agent simply communicates it to every other agent.

The key problem in this process is deciding on the system's control inputs $\mathbf{u}^k$ such that the uncertainty in the state $\mathbf{x}$ is minimized.

5.6.3 Control Parametrization

The objective of the system is to minimize its uncertainty in its joint belief about $\mathbf{x}$. There are many ways to formally define this control problem, the best being a discounted infinite horizon dynamic programming problem (Bertsekas 2005). However, for any relatively complex scenario this becomes intractable and approximate techniques must be used.

Thus, a finite look ahead will be considered and an open loop control policy for each agent developed. To accommodate feedback, a rolling time horizon will be employed. This requires that the control policies be recomputed at short intervals to keep the look ahead approximately constant and allows the system to adapt to changes.

The control policy will be parametrized by a piecewise constant function, defined over N equal time partitions of M time steps each (Goh and Lee 1988). This results in a look ahead of NM time steps. Thus, the open loop control policy for a time interval $[k+1, k+NM]$ can be specified with the parameters $\mathbf{v}_i^k = \{\mathbf{v}_i^k(1), \ldots, \mathbf{v}_i^k(N)\} \in \mathcal{V}_i = (\mathcal{U}_i)^N$ with actual controls given at each time step $k+q$ by $\mathbf{u}_i^{k+q} = \mathbf{v}_i^k(\lceil \frac{q}{M} \rceil)$, where $q \in \{1, \ldots, NM\}$ and $\lceil \cdot \rceil$ represents the roundup operator.

5.6.4 Objective Function

For a given time k, the utility of a future control policy $\mathbf{v}^k = \{\mathbf{v}_1^k, \ldots, \mathbf{v}_p^k\} \in \mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_p$ and observation series $\mathbf{z}^{k+1:k+NM} = \{\mathbf{z}^{k+1}, \ldots, \mathbf{z}^{k+NM}\}$ is proportional to the amount of uncertainty in the resulting posterior belief at time $k+NM$. Actions and observations that produce a smaller uncertainty in the posterior belief are favoured over others. Thus, a suitable cost function may be the posterior entropy:

$$\begin{aligned} C^k(\mathbf{z}^{k+1:k+NM}, \mathbf{v}^k) &= H_{P(\mathbf{x}^{k+NM}|\mathbf{z}^{k+1:k+NM}, \mathbf{s}^{k+1:k+NM}(\mathbf{v}^k), \tilde{\mathbf{Z}}^k, \tilde{\mathbf{S}}^k)} \\ &= -\mathrm{E}_{\mathbf{x}^{k+NM}}\Big[\log P(\mathbf{x}^{k+NM}|\mathbf{z}^{k+1:k+NM}, \mathbf{s}^{k+1:k+NM}(\mathbf{v}^k), \tilde{\mathbf{Z}}^k, \tilde{\mathbf{S}}^k)\Big]. \end{aligned} \quad (5.32)$$

However, the actual observations made will not be known in advance. Thus, an expectation over all possible future observations must be performed, resulting in the expected team cost or objective function:

$$J^k(\mathbf{v}^k) = \mathrm{E}_{\mathbf{z}^{k+1:k+NM}}\big[C^k(\mathbf{z}^{k+1:k+NM}, \mathbf{v}^k)\big]. \quad (5.33)$$

The finite horizon optimal control problem, at time k, becomes the parameter optimization problem:

$$\mathbf{v}^{k*} = \arg\min_{\mathbf{v}^k \in \mathcal{V}} J^k(\mathbf{v}^k). \quad (5.34)$$

For a rolling time horizon, this must be resolved every $N_r \leq NM$ time step.

This parameter optimization problem translates directly to the distributed decision problem discussed earlier and, provided that the objective function is partially separable, can be solved using the decentralized optimization algorithm given in Algorithm 5.1.

5.7 Example: Object Localization

The proposed decentralized control strategy was implemented in a simulated object localization task. For this scenario robots equipped with bearing-only sensors (see Fig. 5.2) and moving in a 2D plane are required to cooperatively localize a collection of stationary point objects.

The multiagent control problem for this type of scenario was previously examined by Grocholsky et al. (2003). In this work each agent shared past observations but developed a plan independently. This section extends the work of Grocholsky by applying the decentralized optimization procedure to find the optimal joint plans.

5.7.1 Modelling

Objects

The state $\mathbf{x}$ is separated into m independent objects; thus

$$\mathbf{x} = \{\mathbf{x}_{o_1}, \ldots, \mathbf{x}_{o_m}\}. \tag{5.35}$$

Since the objects are stationary the time index has been dropped.

Each object o_j is specified by a 2D position $\mathbf{x}_{o_j} = [x_{o_j}, y_{o_j}]^T$ that is independent of all other objects.

Fig. 5.2. Typical mobile robot equipped with a bearing-only sensor (panoramic camera).

Agent Motion

The agents are based on an aircraft model and are described by their position and orientation, $\mathbf{s}_i^k = [x_i^k, y_i^k, \theta_i^k]^T$, travel at a constant velocity $V_i = 50$m/s, and are controlled via a single scalar defining the robots rate of turn $\mathbf{u}_i^k = \dot{\theta}_i^k$. Thus, the deterministic motion model $\mathbf{s}_i^{k+1} = \mathbf{f}_i(\mathbf{s}_i^k, \mathbf{u}_i^{k+1})$ is given by

$$x_i^{k+1} = x_i^k + \frac{2V_i}{\mathbf{u}_i^{k+1}} \sin(\frac{1}{2}\mathbf{u}_i^{k+1}\Delta t) \cos(\theta_i^{k+1} + \frac{1}{2}\mathbf{u}_i^{k+1}\Delta t) \tag{5.36a}$$

$$y_i^{k+1} = x_i^k + \frac{2V_i}{\mathbf{u}_i^{k+1}} \sin(\frac{1}{2}\mathbf{u}_i^{k+1}\Delta t) \sin(\theta_i^{k+1} + \frac{1}{2}\mathbf{u}_i^{k+1}\Delta t) \tag{5.36b}$$

$$\theta_i^{k+1} = \theta_i^k + \mathbf{u}_i^{k+1}\Delta t, \tag{5.36c}$$

where Δt is the time between k and $k+1$.

Observations

It is assumed that each agent i receives an independent bearing observation $\mathbf{z}_{i,o_j}^k$ from each object o_j at each time k; thus $\mathbf{z}_i^k = \{\mathbf{z}_{i,o_j}^k : \forall\, j\}$. The independency assumption allows the observations of the objects to be modelled separately.

The i^{th} agent's observation model for object o_j, defined as the conditional probability density, is assumed to be Gaussian and is given by

$$P(\mathbf{z}_{i,o_j}^k | \mathbf{x}_{o_j}; \mathbf{s}_i^k) = N(\mathbf{z}_{i,o_j}^k; \mathbf{h}_{i,o_j}(\mathbf{x}_{o_j}, \mathbf{s}^k), \mathbf{R}_{i,o_j}^k). \tag{5.37}$$

Here, the notation $N(\xi; \mathbf{m}_\xi, \mathbf{C}_\xi)$ represents a Gaussian (or normal) density defined on the state ξ with a mean of $\mathbf{m}_\xi$ and variance $\mathbf{C}_\xi$.

The mean observation for a given object o_j with state $\mathbf{x}_{o_j}$ when the agent is in state $\mathbf{s}_i^k$ is given by the nonlinear function

$$\begin{aligned} \bar{\mathbf{z}}_{i,o_j}^k &= \mathbf{h}_{i,o_j}(\mathbf{x}_{o_j}, \mathbf{s}_i^k) \\ &= \tan^{-1}\Big(\frac{y_{o_j} - y_i^k}{x_{o_j} - x_i^k}\Big). \end{aligned} \tag{5.38}$$

The variance of the bearing observations is set to $\mathbf{R}_{i,o_j}^k = 25^\circ$ for all agents and objects.

5.7.2 Filtering

Consider some time $k-1$, where the teams belief about $\mathbf{x}_{o_j}$ is Gaussian with mean $\bar{\mathbf{x}}_{o_j}^{|k-1}$ and covariance $\mathbf{P}_{o_j}^{|k-1}$:

$$P(\mathbf{x}_{o_j} | \tilde{\mathbf{Z}}^{k-1}; \tilde{\mathbf{S}}^{k-1}) = N(\mathbf{x}_{o_j}; \bar{\mathbf{x}}_{o_j}^{|k-1}, \mathbf{P}_{o_j}^{|k-1}). \tag{5.39}$$

Here the notation $(\cdot)^{|k-1}$ represents "given information up to $k-1$."

Since all the objects are independent, the belief is simply given by the product of the individual probability densities for each object:

$$P(\mathbf{x}^{k-1}|\tilde{\mathbf{Z}}^{k-1};\tilde{\mathbf{S}}^{k-1}) = \prod_{j=1}^{m} P(\mathbf{x}_{o_j}^{k-1}|\tilde{\mathbf{Z}}^{k-1};\tilde{\mathbf{S}}^{k-1}). \tag{5.40}$$

Thus, only the belief $P(\mathbf{x}_{o_j}^{k-1}|\mathbf{Z}^{k-1};\mathbf{S}^{k-1})$ will be considered.

Owing to the static nature of the objects, the prediction step (corresponding to Eq. (5.29)) can be skipped. If the update step (corresponding to Bayes rule Eq. (5.30)) is implemented directly, due to the nonlinearity in the observation model, the posterior will not be Gaussian (even if the prior is Gaussian). The extended Kalman filter (Maybeck 1982) overcomes this by linearizing the observation model about some nominal state ${}_n\mathbf{x}_{o_j}^k$, given by the prior mean $\bar{\mathbf{x}}_{o_j}^{|k-1}$.

Using a first-order Taylor expansion on $\mathbf{h}_{i,o_j}(\mathbf{x}_{o_j}, \mathbf{s}_i^k)$ about ${}_n\mathbf{x}_{o_j}^k$ yields

$$\mathbf{h}_{i,o_j}(\mathbf{x}_{o_j}, \mathbf{s}_i^k) \approx {}_n\mathbf{z}_{i,o_j}^k + \mathbf{H}_{i,o_j}^k\,[\mathbf{x}_{o_j} - {}_n\mathbf{x}_{o_j}^k], \tag{5.41}$$

where ${}_n\mathbf{z}_{i,o_j}^k = \mathbf{h}_{i,o_j}({}_n\mathbf{x}_{o_j}^k, \mathbf{s}_i^k)$ is the nominal observation and the matrix $\mathbf{H}_{i,o_j}^k = \nabla_{\mathbf{x}}\mathbf{h}_{i,o_j}({}_n\mathbf{x}_{o_j}^k, \mathbf{s}_i^k)$ is the Jacobian of the observation function evaluated at the nominal object state and agent state.

For a bearing observation these become

$${}_n\mathbf{z}_{i,o_j}^k = \tan^{-1}\left(\frac{{}_ny_{o_j}^k - y_i^k}{{}_nx_{o_j}^k - x_i^k}\right) \tag{5.42}$$

and

$$\mathbf{H}_{i,o_j}^k = \frac{1}{{}_nr_{i,o_j}^k}\big[-\sin({}_n\mathbf{z}_{i,o_j}^k),\ \cos({}_n\mathbf{z}_{i,o_j}^k)\big], \tag{5.43}$$

where ${}_nr_{i,o_j}^k = \sqrt{({}_ny_t^k - y_i^k)^2 + ({}_nx_{o_j}^k - x_i^k)^2}$ is the range from the agent to the nominal object state.

Now, with this linearized model the posterior will remain Gaussian for a Gaussian prior. The following update equations take the prior mean and covariance and an observation and produce the associated posterior mean and covariance. It is given in information or inverse covariance form by defining the Fisher information matrix $\mathbf{Y}$ as the inverse covariance, i.e., $\mathbf{Y} \equiv \mathbf{P}^{-1}$,

$$\mathbf{Y}_{o_j}^{|k} = \mathbf{Y}_{o_j}^{|k-1} + \sum_{i=1}^{p}(\mathbf{H}_{i,o_j}^k)^T\,(\mathbf{R}_{i,o_j}^k)^{-1}\,\mathbf{H}_{i,o_j}^k, \tag{5.44}$$

and

$$\begin{aligned}\mathbf{Y}_{o_j}^{|k}\,\bar{\mathbf{x}}_{o_j}^{|k} = {}&\mathbf{Y}_{o_j}^{|k-1}\,\bar{\mathbf{x}}_{o_j}^{|k-1} \\ &+ \sum_{i=1}^{p}(\mathbf{H}_{i,o_j}^k)^T\,(\mathbf{R}_{i,o_j}^k)^{-1}\,\big(\tilde{\mathbf{z}}_{i,o_j}^k - {}_n\mathbf{z}_{i,o_j}^k + \mathbf{H}_{i,o_j}^k\,\bar{\mathbf{x}}_{o_j}^{|k-1}\big).\end{aligned} \tag{5.45}$$

An interesting property of this representation is that the updated or posterior information matrix $\mathbf{Y}_{o_j}^{|k}$ (and hence covariance $\mathbf{P}_{o_j}^{|k}$) is independent of the actual observations, $\mathbf{z}_i^k$, taken [see Eq. (5.44)]. This is an important property and will allow the expected entropy required for the objective function to be calculated very efficiently.

5.7.3 Objective Function

The objective function, defined in Section 5.6.4, represents the expected posterior entropy of the team belief at the end of an NM step time horizon. Since the objects are independent (the density can be decomposed into the product of the densities of each object alone), the entropy becomes a sum of the entropies of each individual object; thus

$$H_{P(\mathbf{x}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})} = \sum_{j=1}^{m} H_{P(\mathbf{x}_{o_j}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})}, \tag{5.46}$$

where the entropy of the Gaussian density for the object state is given by

$$H_{P(\mathbf{x}_{o_j}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})} = -\frac{1}{2}\log\Big((2\pi e)^{d_x}\left|\mathbf{Y}_{o_j}^{|k+NM}\right|\Big), \tag{5.47}$$

with $d_x = 2$ the dimension of the state $\mathbf{x}_{o_j}$.

It is noted that the entropy is only dependent on the information matrix $\mathbf{Y}_{o_j}^{|k+NM}$. By examining the modelling and filtering equations of Section 5.7.2, it can be seen that the observations only influence the covariance or information matrix (hence the posterior entropy) by changing the point at which the observation model is linearized (through changing the prior densities mean).

Thus, to remove this dependency and the requirement to perform the expectation in Eq. (5.33), the nominal state, about which the observation model is linearized, will be given by the mean object state at time k:

$${}_n\mathbf{x}_{o_j}^{k+l} = \bar{\mathbf{x}}_{o_j}^k, \quad \forall\, l \in \{1,\ldots,NM\}. \tag{5.48}$$

Hence, the posterior information matrix will be independent of the observation sequence $\tilde{\mathbf{z}}_{o_j}^{k+1:k+NM}$ and may be evaluated directly using

$$\mathbf{Y}_{o_j}^{|k+NM} = \mathbf{Y}_{o_j}^{|k} + \sum_{i=1}^{p}\sum_{l=1}^{NM} \mathbf{I}_i^{k+l}(\mathbf{v}_i^k), \tag{5.49}$$

where $\mathbf{I}_{i,o_j}^{k+l}(\mathbf{v}_i^k)$ is the observation information matrix and is given by

$$\mathbf{I}_{i,o_j}^{k+l}(\mathbf{v}_i^k) = (\mathbf{H}_{i,o_j}^{k+l})^T\,(\mathbf{R}_{i,o_j}^{k+l})^{-1}\,\mathbf{H}_{i,o_j}^{k+l}. \tag{5.50}$$

The posterior entropy of object o_j is now given by

$$\begin{aligned} J_{o_j}^k(\mathbf{v}^k) &= H_{P(\mathbf{x}_{o_j}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})} \\ &= -\frac{1}{2}\log\Big((2\pi e)^{d_x}\left|\mathbf{Y}_{o_j}^{|k+NM}\right|\Big) \end{aligned} \tag{5.51}$$

and the final team objective function, corresponding to the joint entropy of all objects, is

$$J^k(\mathbf{v}^k) = \sum_{j=1}^{m} J^k_{o_j}(\mathbf{v}^k). \tag{5.52}$$

Partial Separability

For each agent to evaluate the objective function, it requires the sum of the observation information matrices from all other agents and over all steps in the planning horizon. The actual observation models, $\mathbf{H}^k_{i,o_j}$ and $\mathbf{R}^k_{i,o_j}$, and the position of the agents $\mathbf{s}^k_i$ are irrelevant once this information is obtained.

- *Impact Space:* Due to this structure, an impact space can be defined that contains the required matrices for each of the m objects.

$$\mathfrak{I} = \prod_{j=1}^{m} \mathcal{M}^S_{2\times 2}, \tag{5.53}$$

  where $\mathcal{M}^S_{2\times 2}$ is the vector space containing all symmetric 2×2 matrices.
- *Impact Function:* The impact function for an agent i maps a decision onto an element of this space by summing the individual information matrices $\mathbf{I}^{k+l}_{i,o_j}(\mathbf{v}^k_i)$ for all $l \in \{1, \ldots, NM\}$ for each object $j \in \{1, \ldots, m\}$ and, i.e.,

$$\Upsilon_i(\mathbf{v}^k_i) = \Big\{ \sum_{l=1}^{NM} \mathbf{I}^{k+l}_{i,o_j}(\mathbf{v}^k_i) : \forall\, j \in \{1, \ldots, m\} \Big\}. \tag{5.54}$$

- *Composition Operator:* This operator combines impacts from different agents. It is given by matrix addition and simply adds corresponding observation information matrices. Thus, if $\alpha^k_a = \Upsilon_a(\mathbf{v}^k_a)$ and $\alpha^k_b = \Upsilon_b(\mathbf{v}^k_b)$, the composition operator is given by

$$\alpha^k_a * \alpha^k_b = \Big\{ \sum_{l=1}^{NM} \mathbf{I}^{k+l}_{a,o_j} + \sum_{l=1}^{NM} \mathbf{I}^{k+l}_{b,o_j} : \forall\, j \in \{1, \ldots, m\} \Big\}. \tag{5.55}$$

- *Generalized Objective Function:* This function evaluates the cost (expected posterior entropy) of the agents decisions directly from the combined system impact. Consider the combined impact $\alpha^k_T = \bigodot_{i=1}^{p} \Upsilon_i(\mathbf{v}^k_i)$, given as

$$\alpha^k_T = \Big\{ \alpha^k_{T,o_j} : \forall\, j \in \{1, \ldots, m\} \Big\}, \tag{5.56}$$

  where $\alpha^k_{T,o_j} = \sum_{i=1}^{p} \sum_{l=1}^{NM} \mathbf{I}^{k+l}_{i,o_j}$. Now, the generalized objective function $\psi^k(\alpha^k_T)$ is given by duplicate word

$$\psi^k(\alpha^k_T) = -\sum_{j=1}^{m} \frac{1}{2} \log\Big((2\pi e)^{d_x} \Big| \mathbf{Y}^{|k+NM}_{o_j} \Big| \Big), \tag{5.57}$$

where $\mathbf{Y}_{o_j}^{|k+NM}$ is given

$$\mathbf{Y}_{o_j}^{|k+NM} = \mathbf{Y}_{o_j}^{|k} + \alpha_{T,o_j}^{k}. \tag{5.58}$$

5.7.4 Collaborative Control

With the above definition, the collaborative multiagent decision problem becomes

$$\mathbf{v}^{k*} = \arg\min_{\mathbf{v}^k \in \mathcal{V}} \psi^k(\alpha_T^k), \tag{5.59}$$

where $\alpha_T^k = \bigodot_{i=1}^{p} \Upsilon_i(\mathbf{v}_i^k)$, and is solved using Algorithm 5.1.

5.8 Results

5.8.1 Two Agents—Single Object

To demonstrate the workings of the decentralized optimization algorithm, a system comprising two agents observing a single stationary object is considered. The configuration is shown in Fig. 5.3. For this scenario each agent has to decide on a control policy consisting of a single control parameter ($N = 1$) that defines its rate of turn over a planning horizon of 12 s.

The optimal joint plans are found by each agent executing Algorithm 5.1. Although this procedure is designed to allow asynchronous execution, it was executed synchronously. Agents communicated after every local iteration with an imposed communication delay corresponding to three iterations.

As each agent has only a bearing sensor, individually they have a very poor ability to localize the object. However, if they cooperate and observe the object from perpendicular directions they can greatly minimize its position uncertainty. However, there is

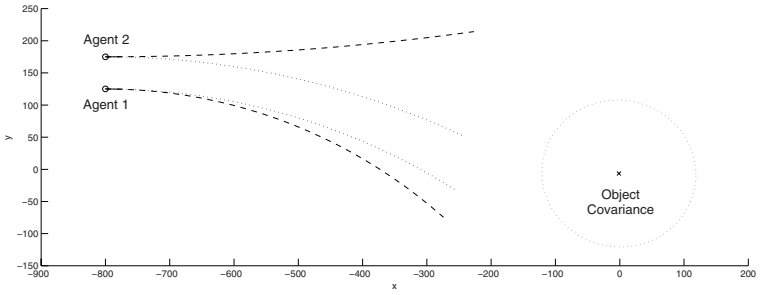


Fig. 5.3. The dashed trajectories corresponding to the jointly optimal plans. The dotted trajectories represent the optimal solution to the corresponding single-agent problem and were used to initialize the negotiated solution. The prior probability density of the position of the object is given by a Gaussian with the mean shown by the cross ($\times$) and the variance by the dotted circle.

also a dependency on the range at which they observe the object, such that a shorter range will give a smaller uncertainty in the measured position.

These seemingly opposing objectives are captured by the single objective function defined in Eq. (5.52). As shown in Fig. 5.3, the optimal decisions cause the agents to move towards the object and separate, such that a better triangulation angle is obtained.

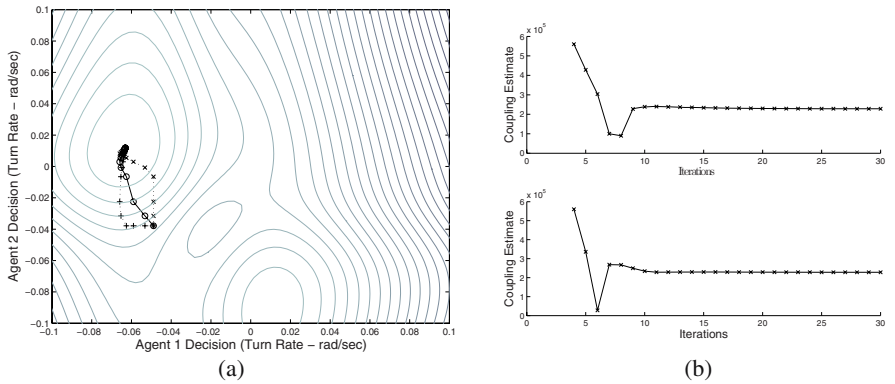
Although this resulting behaviour is intuitive to the observer, an agent cannot reason about this sort of global behaviour. Each agent only knows about the other through the communication of abstract *impacts*; the position of the other agent and its planned trajectory are completely unknown.

Figure 5.4a displays the evolution of the agents' decisions throughout the optimization procedure. Although the communication delay causes a significant difference in the perceived trajectories through the decision space, the system still converges to the optimum. It is noted that one agent never knows about the actual decision of the other; it only knows its impact. Figure 5.4a simply plots the decision corresponding to this impact.

Figure 5.4b plots the interagent coupling as approximated by both agents. The curves have similar shapes, but are not identical because they are measuring the curvature of the objective function at different points in the decision space.

5.8.2 Nine Agents—Eighteen Objects

This scenario consists of nine agents cooperatively localizing 18 objects. The agents start from the left side of Fig. 5.5a, in an arrow formation. They take observations at a rate of 2 Hz and plan a trajectory 16 s into the future (corresponding to an 800-m path). The trajectory is parametrized by four variables defining the required turn rates



(a) (b)

Fig. 5.4. (a) Evolution of agents' decisions through the global decision space $\mathcal{V}$ during the optimization procedure, overlaid with contours of the objective function. The true path $\mathbf{v} = [{}^1\mathbf{v}_1, {}^2\mathbf{v}_2]^T$ is shown with circles ($\circ$), while the pluses ($+$) and crosses ($\times$) represent the perceived path ${}^i\mathbf{v} = [{}^i\mathbf{v}_1, {}^i\mathbf{v}_2]^T$ for agents $i = 1, 2$, respectively. The difference is caused by the communication delays. (b) Coupling estimates ${}^1\hat{K}_{12}$ (top) and ${}^2\hat{K}_{21}$ (bottom) calculated by agents 1 and 2, respectively.

for each quarter. This corresponds to $N = 4$, $M = 8$, and $\Delta t = 0.5\,\mathrm{s}$ and results in an optimal planning problem consisting of 36 parameters distributed over the nine agents.

A rolling planning horizon is employed, requiring that a new plan be developed every second (i.e., $N_r = 2$). When generating the very first plan, the agents initialize their decisions using the locally optimal decision (as discussed in Sect. 5.8.1 for the two-agent case); however at all later stages the decisions are initialized using the previous decision.

The snapshots of the system are shown in Fig. 5.5a–d. Figure 5.5d shows the final state of the system after all the objects have been sufficiently localized, but the current optimal plans are also shown for completeness.

Figure 5.6 shows some data of a typical run of the optimization algorithm. It plots the estimated coupling constants, communication events, and the evolution of the decision parameters from the perspective of a single agent. These data are for agent 6 for the very first decision problem (the results of which are shown in Fig. 5.5a) and demonstrate the relation between coupling (top graph) and communication frequency (middle graph). The agent communicates at a high frequency to agent 5, to which it is coupled

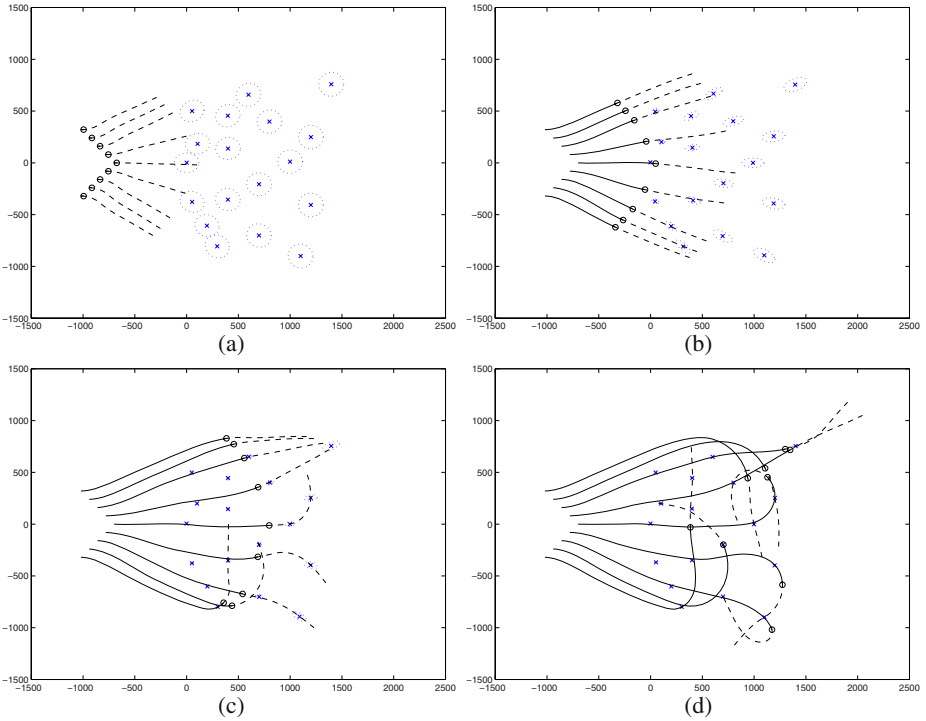


Fig. 5.5. Snapshots throughout the scenario at (a) $k = 0$, (b) $k = 30$, (c) $k = 60$, and (d) $k = 90$. Current agent positions are shown with a circle ($\circ$), and the optimal future planned trajectory with a dotted line. The current probability density of the location of each object is represented by its mean ($\times$) and covariance (dotted circle).

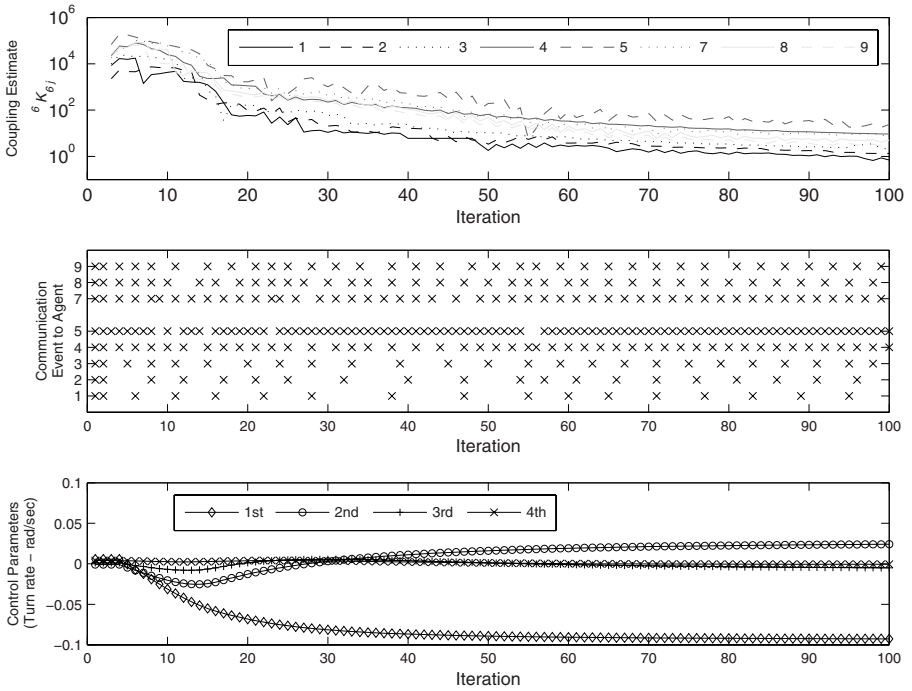


Fig. 5.6. Typical data during a single run of the distributed optimization algorithm. These data correspond to agent 6 for $k = 0$, as seen in to Fig. 5.5a. Top: Estimated coupling terms $^6\hat{K}_{6j}$ for $j \in \{1, 2, 3, 4, 5, 7, 8, 9\}$. Middle: Each cross indicates the time a message is sent to each specific agent in the system (the frequency of these events is determined by the coupling metric, according to Eq. (5.19). Bottom: Evolution of the agent's decision parameters (corresponding to the agent's rate of turn during the planning horizon).

the most, and at a much lower frequency to other agents (especially agent 1), where the interagent coupling is smaller.

Figure 5.7 displays the interagent coupling for the whole system for each snap shot in Fig. 5.5. The i^{th} row of each matrix represents the average of $^i\hat{K}_{ij}$ over all the iterations of Algorithm 5.1 for every other agent j. As expected, the matrix is reasonably symmetric (the coupling terms correspond to cross derivatives, which by definition are symmetric) and shows a large amount of structure. The matrix in Fig. 5.7a displays the intuitive result that agents close to each other are highly coupled (due to the diagonal structure).

However, agent separation is not directly important; the coupling essentially measures the sensitivity of the effect of the information that one agent receives from its observations on the decisions of another. This is demonstrated in the last matrix corresponding to Fig. 5.5d. At this time in the mission all the objects are well localized and for the agents to gather more information (and reduce the uncertainty) about the positions of the objects, they must travel very close to them. Thus, only agents with

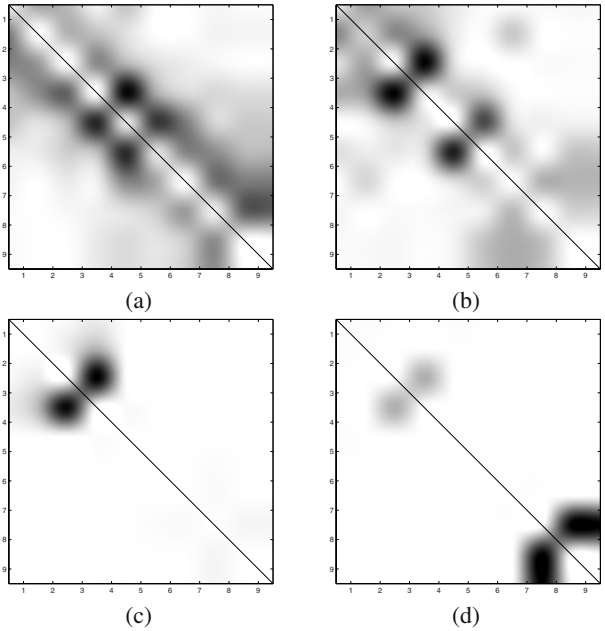


Fig. 5.7. Coupling matrix for the initial decision problem. The checker board type appearance (especially the two minor diagonals) represent that generally agents closer to each other are more strongly coupled (see Fig. 5.5(a)).

planned trajectories passing by a common object are coupled, e.g., agents 8 and 9 and agents 3 and 4.

This coupling metric captures how the underlying agent models interact through the system's objective function, which is precisely what is important for a multiagent system undertaking a specific task.

5.9 Discussion and Future Work

This chapter has approached the problem of multiagent decision making and planning using the tools of asynchronous distributed optimization. This analytical approach led to the definition of a coupling metric that intuitively links the rate at which an agent may refine its local decision to its interagent communication frequency and transmission delays. The coupling is determined by the cross derivatives of the objective function and captures how the underlying agent models interact with the definition of the system's task.

This decentralized negotiation algorithm was used to control a simulated multiagent system involving multiple mobile robots undertaking an object localization task. This example demonstrated that agents are required to communicate often only with the agents to which they are coupled.

It is envisaged for much larger distributed systems that sparseness of inter-agent coupling (e.g., as shown in Fig. 5.7) will be more prevalent, causing each agent to be coupled only to a small subset of the system. This will overcome the main requirement that each agent must negotiate with every other agent, as only agents that are coupled are required to communicate.

Another extension under investigation is the use of hierarchical clustering techniques, such as described in Balch (2000), to build and maintain dynamic hierarchies within the system. This can be illustrated by considering the coupling structure shown in Fig. 5.7b, which can be easily partitioned into three clusters of $\{1, 2, 3, 4\}$, $\{5, 6\}$, and $\{7, 8, 9\}$. Agents within a cluster are highly coupled and communicate directly to each other, while intercluster communication is directed via a cluster head, which summarizes the impacts of all the agents in the cluster. This abstraction of agents to clusters can be taken to higher levels with clusters of clusters and clusters of clusters of clusters, and so forth, each participating in higher-level negotiations.

Acknowledgements

This work is partly supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government, and by CSIRO Industrial Physics.

References

Balch, T. (2000). Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous Robots*, Belmont, Massachusetts 8(3):209–138.

Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control*. Athena Scientific.

Cover, T. M., and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley, New York.

Furukawa, T., Dissanayake, G., and Durrant-Whyte, H. F. (2003). Time-optimal cooperative control of multiple robot vehicles. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14–19, 2003, Taipei, Taiwan*, pages 944–950. IEEE.

Goh, C., and Teo, K. (1988). Control parametrization: a unified approach to optimal control problems with general constraints. *Automatica*, 24(1):3–18.

Grocholsky, B., Makarenko, A., Kaupp, T., and Durrant-Whyte, H. F. (2003). Scalable control of decentralised sensor platforms. In Zhao, F., and Guibas, L. J., editors, *Information Processing in Sensor Networks, Second International Workshop, IPSN 2003, Palo Alto, CA, April 22–23, 2003, Proceedings*, volume 2634 of *Lecture Notes in Computer Science*, pages 96–112. Springer, New York.

Liggins, M.E., I., Chong, C.-Y., Kadar, I., Alford, M., Vannicola, V., and Thomopoulos, S. (1997). Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85(1): 95–107. New York.

Manyika, J., and Durrant-Whyte, H. F. (1994). *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Ellis Horwood, New York.

Mathews, G. M., Durrant-Whyte, H. F., and Prokopenko, M. (2006). Scalable decentralised decision making and optimisation in heterogeneous teams. In *Proccedings of IEEE International Conference on Multi-Sensor Fusion and Integration for Intelligent Systems (MFI-2006), Heidelberg, Germany*, pages 383–388. IEEE.

Maybeck, P. S. (1979–1982). *Stochastic Models, Estimation and Control*. Academic, New York.

Tsitsiklis, J. N., Bertsekas, D. P., and Athens, M. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, AC-31:803–812.

6

Learning Mutation Strategies for Evolution and Adaptation of a Simulated Snakebot

Ivan Tanev

6.1 Introduction

Wheelless, limbless snakelike robots (Snakebots) feature potential robustness characteristics beyond the capabilities of most wheeled and legged vehicles—an ability to traverse terrain that would pose problems for traditional wheeled or legged robots and insignificant performance degradation when partial damage is inflicted. Some useful features of Snakebots include smaller size of the cross-sectional areas, stability, ability to operate in difficult terrain, good traction, high redundancy, and complete sealing of the internal mechanisms (Hirose 1993; Dowling 1997).

Robots with these properties open up several critical applications in exploration, reconnaissance, medicine, and inspection. However, compared to the wheeled and legged vehicles, Snakebots feature more difficult control of locomotion gaits and inferior speed characteristics. In this work we address the following challenge: how to automatically develop control sequences of Snakebot's actuators that allow for achieving the fastest possible speed of locomotion.

In principle, the task of designing the code of a Snakebot could be formalized and the formal mathematical models could be incorporated into direct programmable control strategies. However, the eventual models would feature enormous complexity and would not be recognized as having a known analytically exact optimal solution. The complexity of the model stems from the large number of degrees of freedom of the Snakebot, which cannot be treated independently of one another. The dynamic patterns of position, orientation, velocity vectors, and the points and instances of contact with the surface (and, consequently, the vectors of resulting traction forces, that propel the Snakebot) of each of the Snakebot is morphological segments have to be considered within the context of other segments. Furthermore, often the dynamic patterns of these parameters cannot be deterministically inferred from the desired velocity characteristics of the locomotion of the Snakebot. Instead, its locomotion is viewed as an emergent property at a higher level of consideration of a complex hierarchical system, comprising many relatively simple entities (morphological segments). Higher-level properties

of the system as a whole and the lower-level properties of entities it comprises cannot be induced from each other. Owing to their ability ot find a near-optimal solution in a reasonable runtime, evolutionary approaches (Takamura et al. 2000; Mahdavi and Bentley 2003) are considered efficient ways of tackling such ill-posed problems.

As an instance of evolutionary algorithms, Genetic Algorithms (GA) differ from Genetic Programming (GP) mainly in the genotypic representation (i.e., the chromosome) of potential solutions (Goldberg 1989). Instead of representing the solution as a computer program (usually a parse tree) featuring arbitrary structure and complexity as in GP, a GA employs a fixed-length linear chromosome. This difference implies a favorable computational advantage of the GA over GP for simple problems because linear chromosomes are computationally can be manipulated and interpreted more efficiently. For complex tasks, however, such as evolution of locomotion gaits of a Snakebot, the runtime overhead associated with manipulation of the genotype is negligible compared to the more significant overhead of the fitness evaluation of evolved (either simulated or real) artifacts. Moreover, an efficient GA (in terms of computational effort or number of fitness evaluations) often requires incorporation of computationally heavy probabilistic learning models (Pelikan et al. 1999) aimed at discovering and maintaining the complex interrelations between variables in the genome. In addition, the fixed-length genome usually implies that the latter comprises various carefully encoded domain-dependent parameters of the solution with an a priori known structure and complexity. This might be a concern if no such knowledge is available in advance, but rather has to be automatically and autonomously discovered by the evolving artifact. The latter is especially true when the artifact has to perform in unanticipated, uncertain environmental conditions or under its own (possibly degraded) mechanical abilities.

An example of the successful implementation of evolution (using a GA) and adaptation (via hierarchical, two-layered Q-learning) of locomotion gaits of a real-world snakelike robot was demonstrated by Ito et al. (2003). Each gene in the linear chromosome represents the angle of the corresponding joint in the snake, and the number of genes is the same as the number of joints. This work demonstrates the efficiency of a canonical GA for the particular complexity of the task—evolution of two-dimensional gaits of a snake having five joints. The efficiency of the GA is adequate even without the need to consider either the scalability problem (the inflation of search space with the increase in the number of joints) or the linkage problem (the correlation between the genes in linear chromosomes). Several similar methods of adaptation combining evolutionary algorithms (either GA or GP for off-line evolution of a model of the artifact) and learning (for on-line adaptation of the real artifact) have recently been proposed (Kimura et al. 2001; Kamio et al. 2003). The learning component in these approaches is usually intended to tune in on-line and mode the solution obtained off-line via simulated evolution. Such approaches as an on-line parametric optimization of a solution via a local search are efficient when changes to the fitness landscape are assumed to be insignificant. However, adaptation to unanticipated, unknown, or uncertain changes in fitness landscapes might require the discovery of completely new solutions, which often could not be achieved by parametric optimization of already existing solutions. We assume that GP alone, which is able to balance inherently exploration

(of completely new solutions) and exploitation (of previously obtained solutions) by maintaining a diverse population of solutions, offers good opportunities to discover these new solutions.

An inverse approach, based on evolution (estimation) of the morphology of a potentially damaged robot given a controller (instead of evolving a controller given a morphology of the damaged robot) allows one to evolve a damage hypothesis after failure and then to reevolve a compensatory neural controller to restore the robots functionality (Bongard and Lipson 2004). Conversely, in our work we adhere to the conventional approache of employing simulated evolution to develop the compensatory traits of a controller given the unanticipated changes of morphology due to partial damage of the Snakebot.

The objectives of our work are: (i) to explore the feasibility of applying GP for efficient automatic design of the fastest possible locomotion of realistically simulated Snakebot, and (ii) to investigate the adaptation of such locomotion to challenging environment and degraded abilities (due to partial damage) of simulated Snakebot. In Tanev and Ray (2005) we discussed the feasibility of applying an evolutionary approach for automated development of locomotion gaits of Snakebots. Later we demonstrated the evolution of nonperiodic postures of the Snakebot and verified the versatility of genetic programming for evolution and adaptation to environmental challenges and damages (Tanev et al. 2005). In this work we discuss the biologically plausible (Caporale 2003; Kirschner and Gerhart 2005) nonrandom mutations implemented through learning mutation strategies (LMS) in GP. We are especially interested in the implications of LMS on the efficiency of evolution and adaptation of the Snakebot.

The approach we present for incorporating LMS is implemented via learning probabilistic context-sensitive grammar (LPCSG), employed to express the preferable syntactical bias of mutation operation in GP. LPCSG is related to approaches that use a grammar to define the allowed syntax of the evolved artifacts. Examples of such approaches are grammatical evolution (GE) (O'Neill and Ryan 2003), grammar-based genetic programming (Wong 2005), and grammar-based genetic algorithms (Antonisse 1991). The genotype evolved via GE encodes the sequence of grammar rules that should be applied during the simulated gene expression phase in order to generate the phenotype.

From another perspective, our work is also related to the incorporation of the estimation of distribution algorithms (EDA) for biased mutations in evolutionary computations, mostly in GA (Pelikan et al. 1999). Motivated by the demonstrated advantages of both the GE and EDA in GA, we intend to merge the two approaches in a way that allows for the biased mutation in GP (rather than in GA, as in EDA) to be implemented via adjustable, learned preferences (rather than "hard coded" in the chromosome, as in GE) in choosing the appropriate rule from the set of alternative, potentially applicable grammar rules. Although a few grammar-based EDAs have recently been proposed (Bosman and de Jong 2004; Shan et al. 2004), in neither of these methods has the incorporation of LPCSG in GP been explored.

The remainder of the chapter is organized as follows. Section 6.2 emphasizes the main features of GP proposed for evolution of locomotion of the Snakebot. Section 6.3 introduces the proposed approach of incorporating LPCSG in GP. Section 6.4 presents

the empirically obtained results of efficiency of evolution and adaptation of the Snakebot to challenging environment and partial damage. Section 6.5 discusses an alternative, interactive mechanism for learning the mutation strategies. Section 6.5 draws a conclusion.

6.2 Genetic Programming for the Design of Snakebot Gaits

6.2.1 Morphology of the Snakebot

The Snakebot is simulated as a set of identical spherical morphological segments ("vertebrae") linked together via universal joints. All joints feature identical (finite) angle limits and each joint has two attached actuators ("muscles"). In Snakebot's initial, standstill position the rotation axes of the actuators are oriented vertically (vertical actuator) and horizontally (horizontal actuator) and rotate the joint in the horizontal and vertical planes, respectively. In view of the representation of the Snakebot, the task of designing the fastest locomotion can be rephrased as developing temporal patterns of desired turning angles of horizontal and vertical actuators of each segment that result in its fastest overall locomotion. The proposed representation of the Snakebot as a homogeneous system comprising identical morphological segments is intended to significantly reduce the size of the search space of the GP. Moreover, because the size of the search space does not necessarily increase with an increase in the Snakebot's complexity (i.e., the number of morphological segments), the proposed approach allows achievement of favorable scalability characteristics of GP.

In the representation considered, we use a pyramid approximation of the Coulomb isotropic friction model. No anisotropic (directional) friction between the morphological segments and the surface is considered. Despite the anticipated ease of simulation and design of eventual morphological segments featuring anisotropic friction with the surface [using passive wheels (Hirose 1993; Ito et al. 2003) or "belly" scales], such an approach would have the following drawbacks:

1. Wheels attached to the morphological segments are mainly effective in two-dimensional locomotion gaits. However, neither the fastest gaits in unconstrained environments nor the adaptive gaits in challenging environments (narrow tunnels, obstacles, etc.) are necessarily two dimensional. In three-dimensional locomotion gaits the orientation (the pitch, roll, and yaw angles) of morphological segments at the instant of contact with the surface is arbitrary, which renders the design of effective wheels for such locomotion gaits a nontrivial engineering task.
2. Wheels may compromise the potential robustness characteristics of the Snakebot because they can be trapped easily in the challenging environments (rugged terrain, obstacles, etc.).
3. Wheels potentially reduce the application areas of the Snakebot because their engineering design implies lack of complete sealing of all mechanisms.
4. Belly scales would not promote any anisotropic friction if the Snakebot operates on a smooth, flat, clean, and/or loose surface. Therefore the generality of locomotion gaits and their robustness with respect to various environmental conditions would be compromised.

5. Belly scales are efficiently utilized as a source of anisotropic friction in some locomotion gaits of natural snakes. However, these gaits usually involve a large number of complex muscles located immediately under the skin of the snake to lift the scales off the ground, angle them forward, and then push them back against the surface. In the Snakebot, implementing actuators that mimic such muscles of natural snakes would be too expensive and thus infeasible from an engineering point of view.

6.2.2 Genetic Programming

Algorithmic Paradigm

GP (Koza 1992) is a domain-independent problem-solving approach in which a population of computer programs (individuals' genotypes) is evolved to solve problems. The simulated evolution in GP is based on the Darwinian principle of reproduction and survival of the fittest. The fitness of each individual is based on the quality of the performance of the phenotype of the simulated individual in a given environment.

Set of Functions and Terminals

In applying GP to the evolution of Snakebot, the genotype is associated with two algebraic expressions that represent the temporal patterns of desired turning angles of both the horizontal and vertical actuators of each morphological segment. Because locomotion gaits, by definition, are periodical, we include the periodic functions `sin` and `cos` in the function set of GP in addition to the basic algebraic functions. Terminal symbols include the variables `time`, `index` of the segment of the Snakebot, and two constants: `Pi` and `random` within the range $[0, 2]$. The main parameters of the GP are shown in Table 6.1.

The introduction of variable `time` reflects our objective to develop *temporal* patterns of turning angles of actuators. The choice of the trigonometric functions `sin` and `cos` reflects our intention to verify the hypothesis, as first expressed by Miturich in the 1920s (Andrusenko 2001), that undulate motion mechanisms can yield efficient gaits of snakelike artifacts operating in air, land, or water.

Table 6.1. Main parameters of GP

Category	Value
Function set	{sin, cos, nop, +, -, *, /}
Terminal set	{time, segment_ID, Pi, random constant, ADF}
Population size	200 individuals
Selection	Binary tournament, selection ratio 0.1, reproduction ratio 0.9
Elitism	Best four individuals
Mutation	Random subtree mutation, ratio 0.01
Fitness	Velocity of simulated Snakebot during the trial
Trial interval	180 time steps, each time step accounting for 50 ms of "real" time
Termination criterion	(Fitness $>$100) *or* (Generations$>$ 40)

From another perspective, by introducing the trigonometric functions we are attempting to mimic (at functional, rather than neurological level) the central pattern generator (CPG) in the central nervous system. The CPG, usually located in the ganglia or spinal cord of animals, is believed to be necessary and sufficient for the generation of rhythmic patterns of activities (Levitan and Kaczmarek 2002). CPG for robot control typically comprises coupled neural controllers, which generate (without the need for external feedback) the motion pattern of actuators in the respective morphological segments of the artifact. The approach of employing CPG for developing the locomotion gaits of the Snakebot would be based on an iterative process (e.g., employing the machine learning and/or evolutionary computation paradigms) of tuning the main parameters of CPG, including, e.g., the common single frequency across the coupled oscillators, the fixed-phase relationship between the oscillators, and the amplitude of each oscillation. The proposed approach of applying GP for evolution of locomotion gaits of the Snakebot shares some of the features of CPG-based approaches, such as the open-loop, sensorless control scheme and the incorporation of coupled oscillators. Unlike the CPG-based approaches, however, the proposed method incorporates too little domain-specific knowledge about the task.

The rationale of employing automatically defined functions (ADF) is based on the empirical observation that although the straightforward, independent encoding of the desired turning angles of both horizontal and vertical actuators allows GP to adequately explore the huge search space and, ultimately, to discover the areas that correspond to fast locomotion gaits in solution space, the evolvability of such encoding is relatively poor. We discovered that: (i) the motion patterns of horizontal and vertical actuators of each segment in fast locomotion gaits are highly correlated (e.g., by frequency, direction, etc.), and (ii) discovering and preserving such correlation by GP is associated with enormous computational effort. ADF is employed in our approach as a way of limiting the search space by introducing modularity and reuse of the evolved code to allow GP to explicitly evolve the correlation between motion patterns of horizontal and vertical actuators in a form of shared fragments in algebraic expressions of desired turning angles of the actuators. Moreover, we observed that the best result is obtained by: (i) allowing the use of ADF as a terminal symbol in the algebraic expression of desired turning angle of only the vertical actuator, and (ii) evaluating the value of ADF by making it equal to the value of the currently evaluated algebraic expression of the desired turning angle of the horizontal actuator.

Context-Free Grammar for Canonical GP

The context-free grammar (CFG), G, usually employed to define the allowed syntax of individuals in GP consists of (N, Σ, P, S), where N is a finite set of nonterminal symbols, Σ is a finite set of terminal symbols that is disjoint from N, S is a symbol in N that is indicated as the start symbol, and P is a set of production rules, where a rule is of the form

$$V \rightarrow w,$$

with V a nonterminal symbol and w a string consisting of terminals and/or nonterminals. The term "context-free" comes from the feature that the variable V can always

be replaced by w, no matter in what context it occurs. The set of nonterminal symbols of G of GP employed to develop the temporal patterns of desired turning angles of horizontal and vertical actuators of segments that result in fastest overall locomotion of the Snakebot is defined as follows:

$$\texttt{N} = \{\texttt{GP, STM, STM1, STM2, VAR, CONST\_x10, CONST\_PI, OP1, OP2}\},$$

where STM is a generic algebraic statement, STM1 is a generic unary (e.g., sin, cos, nop) algebraic statement, STM2 is a generic binary (dyadic, e.g. $+$, $-$, $*$, and $/$) algebraic statement, VAR is a variable, OP1 is a unary operation, OP2 is a binary (dyadic) operation, CONST_x10 is a random constant within the range $[0 \ldots 20]$, and CONST_PI equals either 3.1416 or 1.5708. The set of terminal symbols is defined as

$$\Sigma = \{\texttt{sin}, \texttt{cos}, \texttt{nop}, +, -, *, /, \texttt{time}, \texttt{segment\_id}\}$$

where sin, cos, nop, $+$, $-$, $*$ and $/$ are terminals that specify the functions in the generic algebraic statements. The start symbol is GP, and the set of production rules expressed in Backus-Naur form (BNF) is as shown in Fig. 6.1. GP uses the defined production rules of G to create the initial population and to mutate genetic programs. In the canonical GP, the production rules with multiple alternative right-hand sides (such as rules 2, 4, 6, 7, and 9 shown in Fig. 6.1) are usually chosen *randomly* during these operations.

Fitness Evaluation

The fitness function is based on the velocity of the Snakebot, estimated from the distance that its the center of mass travels during the trial. Fitness of 100 (the one of termination criteria shown in Table 6.1) is equivalent to a velocity that displaces the Snakebot a distance equal to twice its length.

```
(1)         GP ⟶ STM
(2.1-2.5)   STM ⟶ STM1|STM2|VAR|CONST_x10|CONST_PI
(3)          STM1 ⟶ OP1 STM
(4.1-4.6)     OP1 ⟶ sin|cos|nop|-|sqr|sqrt
(5)          STM2 ⟶ OP2 STM STM
(6.1-6.4)     OP2 ⟶ +|-|*|/
(7.1-7.2)    VAR  ⟶ time|segment_id
(8)          CONST_x10 ⟶ 0..20
(9.1-9.2)    CONST_PI  ⟶ 3.1416|1.5708
```

Fig. 6.1. BNF of production rules of the context-free grammar G of GP, employed for automatic design of locomotion gaits of the Snakebot. The following abbreviations are used: STM—generic algebraic statement, STM1—unary algebraic statement, STM2—binary (dyadic) algebraic statement, VAR—variable, OP1—unary operation, and OP2—binary operation.

Representation of Genotype

Inspired by its flexibility and the recently emerged widespread adoption of document object model (DOM) and extensible markup language (XML) (Bray et al. 2000), we represent the evolved genotypes of the Snakebot as DOM parse trees featuring equivalent flat XML text. Both the calculation of the desired turning angles during fitness evaluation and the genetic operations are performed on DOM parse trees via API of the off-the-shelf DOM parser.

Genetic Operations

Selection is a binary tournament. Crossover is defined in a strongly typed way in that only the DOM nodes (and corresponding DOM subtrees) of the same data type (i.e., labeled with the same tag) from parents can be swapped. The subtree mutation is allowed in a strongly typed way in that a random node in the genetic program is replaced by a syntactically correct subtree. The mutation routine refers to the data type of the currently altered node and applies the chosen rule from the set of applicable rewriting rules as defined in the grammar of GP. The selection of the grammar rule that should be applied to the currently altered tree node during the mutation is *random* in the canonical implementation of GP and *biased* in the proposed approach of applying LMS, as is elaborated in Section 6.3.

Open Dynamic Engine

We have chosen Open Dynamics Engine (ODE) (Smith 2006) to provide a realistic simulation of the physics of applying forces to phenotypic segments of the Snakebot. ODE is a free, industrial quality software library for simulating articulated rigid body dynamics. It is fast, flexible, and robust and has built-in collision detection.

6.3 Incorporating LMS in GP

6.3.1 Learning Probabilistic Context-Sensitive Grammar

The proposed approach is based on the idea of introducing bias in applying the most preferable rule from among the grammar rules with multiple, alternative right-hand sides (RHS). We presume that the preferences of applying certain production rules depend on the surrounding grammatical context, defining which rules have been applied before. The probability distributions (PD) $p_1^i, p_2^i, \ldots, p_N^i$ for each $context_i$ for each grammar rule with multiple RHS are initially uniform, and then learned (tuned) incrementally at each generation from the subset of the best-performing Snakebots. The learned PD is then used as a bias to steer the mutation of the Snakebots.

In the proposed approach, the learning probabilistic context-sensitive grammar (LPCSG), G^*, is proposed as a formal model describing such mutations. G^* is introduced as a set of the same attributes (N^*, Σ^*, P^*, S^*) as the CFG G defined in Section 2.2. The attributes N^*, Σ^*, and S^* are identical to the corresponding attributes N, Σ, and S of G. The set of production rules P^* of G^* is derived from P of G as follows:

1. Production rules of P_S ($P_S \subset P$) of G that have a single right-hand side are defined in the same way in $P*$ as in P.
2. Production rules in P_M ($P_M \subset P$) of G that feature multiple right-hand side alternatives $V \rightarrow w_1|w_2|\ldots|w_N$ are redefined for each instance i of the context as follows:

$$\begin{aligned}
&\textit{context}_i V \rightarrow \textit{context}_i\ w_1 \quad (p_1^i)\\
&\textit{context}_i V \rightarrow \textit{context}_i\ w_2 \quad (p_2^i)\\
&\ldots\\
&\textit{context}_i V \rightarrow \textit{context}_i\ w_N \quad (p_N^i),
\end{aligned}$$

where $p_1^i, p_2^i, \ldots, p_N^i$ ($\sum_1^N p_n^i = 1$) are the probabilities of applying each alternative rule with the left-hand side nonterminal V for the given $\textit{context}_i$.

Applying the `IF-THEN` stimulus-response paradigm, which usually expresses the reactive behavioral strategies of intelligent entities in AI (e.g., software agents, robots, etc.) to such biased mutation operations in GP and viewing the evolved genotype not only as an evolving, but also as a learning intelligent entity, we can model the above sample rule of G^* by the following behavioral `IF-THEN` statement:

The LMS strategy in our approach comprises the dynamic set of `IF-THEN` rules created and tuned by parsing the syntax of the best-performing Snakebots of the current generation. A sample of biased application of production rules of G^* according to the

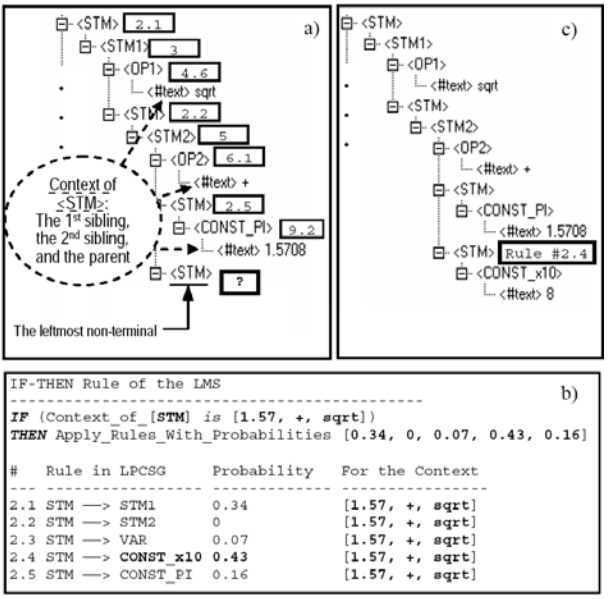


Fig. 6.2. Sample of biased application of production rules of G^*: the current leftmost nonterminal, as shown in (a) is `STM`, which requires applying one of the production rules 2.1–2.5 (refer to Fig. 6.1). For the considered context (a), the LMS of applying rules 2.1-2.5 (b) suggests a highest probability for applying the production rule 2.4, yielding the genetic program as shown in (c).

if `Context_of_[`V`] is [`$context_i$`]` **then**
`Apply_Rules_With_Probabilities(`$p_1^i, p_2^i, \ldots p_N^i$`).`

learned PD and the corresponding `IF-THEN` rule of LMS for the considered leftmost nonterminal and the context are shown in Fig. 6.2.

6.3.2 GP Incorporating LMS

The principal steps of the algorithm of GP that incorporates LMS via LPCSG are shown in Fig. 6.3. As the figure illustrates, additional `Steps 6` and `9` are introduced in the canonical algorithm of GP. The LMS is updated on `Step 6`, and the new offspring, created by applying the proposed biased mutation via LPCSG on `Step 9`, are inserted into the already reproduced—via canonical crossover (`Step 7`) and mutation (`Step 8`)—growing new population of Snakebots. The parameter K_{LMS} defines the ratio of the number of offspring $\#\mathrm{N}_{LMS}$ created via biased mutation using LMS and the number of offspring $\#\mathrm{N}_{CO}$ created via canonical crossover. K_{LMS} is dynamically tuned on `Step 6` based on the stagnation counter C_S, which maintains the number of most recent generations without improvement of the fitness value. In our implementation, K_{LMS} is kept within the range $[0, 5]$, and is defined according to the following rule:

$$\mathrm{K}_{LMS} = 5 - \texttt{smaller\_of}(5, \mathrm{C}_S)$$

Lower values of K_{LMS} in stagnated population (i.e., for $\mathrm{C}_S > 0$) favor reproduction via canonical random genetic operations over reproduction using biased mutation via LMS. As we investigated empirically, the low values of K_{LMS} facilitate avoiding premature convergence by increasing the diversity of the population and, consequently, accelerating the escape from the (most likely) local optimal solutions discovered by the steering bias of the current LMS. Conversely, replacing the usually random genetic operations of canonical GP with the proposed biased mutation when K_{LMS} is close to its maximum value (i.e., for $\mathrm{C}_S = 0$) can be viewed as a mechanism for growing and preserving the proven beneficial building blocks in evolved solutions rather than destroying them by the usual random crossover and mutation.

Updating (Fig. 6.3, `Step 6`) and applying LMS during the biased mutation (Fig. 6.3, `Step 9`) implies maintaining a table that represents the set of learned `IF-THEN` rules. Each entry in the table stores the context, the left-hand side nonterminals, the list of right-hand side symbols, the aggregated reward values, and the calculated probability of applying the given production rule for the given context. A new entry is added or the aggregated reward value of an existing entry is updated by extracting the syntactic features of the best-performing genetic programs (the mating pool) of the current generation. The outdated entries, added four or more generations before are deleted, keeping the total number of entries in the table between 300 and 500. The string of characters, comprising the right-hand side `RHS` of given production rule that should be applied to the current leftmost nonterminal (i.e., the corresponding

```
Step 0:   Creating Initial Population and Clearing PDD;
Step 1:   While (true) do begin
Step 2:       Evaluating Population;
Step 3:       Ranking Population;
Step 4:       if Termination Criteria then Go to Step 10;
Step 5:       Selecting the Mating Pool;
Step 6:       Updating LMS and K_LMS;
Step 7:       Creating #N_CO offspring via canonical crossover;
Step 8:       Mutating current population via canonical mutation;
Step 9:       Creating #N_LMS offspring via mutation of mating pool using LMS;
Step 10:  end;
```

Fig. 6.3. Algorithm of GP incorporating LMS. `Steps 6` and `9` are specific for the proposed approach. `Steps 0, 2--5, 7`, and `8` are common principal steps of canonical GP.

left-hand symbol in production rule, `LHS`) for the given context C is obtained by the function `GetProduction([in] C, [in] LHS, [out] RHS)`, which operates on the LMS table as shown in Fig. 6.4.

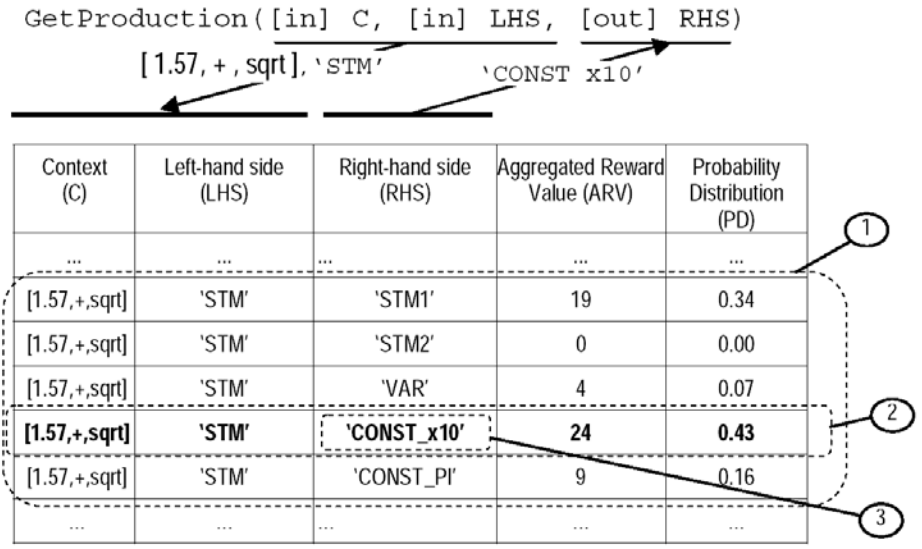


Fig. 6.4. Obtaining the most preferable right-hand side (`RHS`) of production rule of LPCSG that should be applied to the leftmost nonterminal (i.e., left-hand symbol, `LHS`), and the context (C) according to a sample `IF-THEN rule` of the current LMS: (1) Selecting the set of entries associated with the entries featuring the given `LHS` and `C`, (2) Choosing an entry from the obtained result set with a probability proportional to the learned PD, and (3) returning the RHS of the chosen production rule. The sample `IF-THEN` rule of the LMS shown here is the same as depicted in Fig. 6.2.

6.4 Results

This section discusses empirically obtained results verifying the effects of incorporating LMS on the efficiency of GP applied for the following two tasks: (i) *evolution* of the fastest possible locomotion gaits of the Snakebot for various fitness conditions, and (ii) *adaptation* of these locomotion gaits to challenging environment and degraded mechanical abilities of the Snakebot. These tasks, considered as relevant for successful accomplishment of anticipated exploration, reconnaissance, medicine, or inspection missions, feature different fitness landscapes. Therefore, the experiments discussed in this section are intended to verify the versatility and the scope of applicability of the proposed approach. In all of the cases considered, the fitness of the Snakebot reflects its low-level objective (i.e., *what* is required to be achieved) in these missions, namely, to be able to move fast regardless of environmental challenges or degraded abilities. The experiments discussed illustrate the ability of the evolving Snakebot to learn *how* (e.g., by discovering beneficial locomotion traits) to accomplish the required objective without being explicitly taught about the means to do so. Such *know-how* acquired by the Snakebot automatically and autonomously can be viewed as a demonstration of emergent intelligence (Angeline 1994), in that the task-specific knowledge of *how* to accomplish the task emerges in the Snakebot from the interaction of the problem solver and the fitness function.

6.4.1 Evolution of Fastest Locomotion Gaits

Figure 5 shows the results of the evolution of locomotion gaits for cases where fitness is measured as velocity in any direction. Despite the fact that fitness is unconstrained and measured as Snakebot's velocity in *any* direction, *sidewinding* locomotion (defined as locomotion predominantly perpendicular to its long axis) emerged in all ten independent runs of GP, suggesting that it provides superior speed characteristics for the Snakebot's considered morphology. As Fig. 6.5c illustrates, incorporating LMS in GP is associated with computational effort (required to achieve probability of success 0.9) of about 20 generations, which is about 1.6 times faster than canonical GP with CFG. Sample snapshots of evolved best-of-run sidewinding locomotion gaits are shown in Fig. 6.5d–g.

In order to verify the superiority of velocity characteristics of sidewinding, we compared the fitness convergence characteristics of evolution in an unconstrained environment for the following two cases: (i) unconstrained fitness measured as velocity in any direction (as discussed above and illustrated in Fig. 6.5), and (ii) fitness measured as velocity in the forward direction only. The results of evolution of forward (rectilinear) locomotion, shown in Fig. 6.6, indicate that nonsidewinding motion, compared to sidewinding, features much inferior velocity characteristics. The results also demonstrate that GP with LMS on average converges almost four times faster and to higher values than canonical GP. Snapshots taken during the motion of a sample evolved best-of-run sidewinding Snakebot are shown in Fig. 6.6c and d.

The results of evolution of rectilinear locomotion of a simulated Snakebot confined in a narrow "tunnel" are shown in Fig. 6.7. As the fitness convergence characteristics of

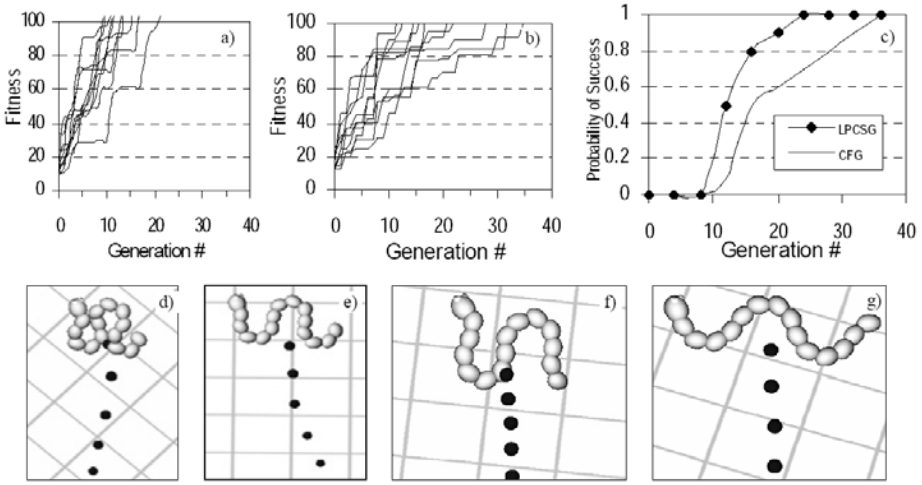


Fig. 6.5. Evolution of locomotion gaits for cases where fitness is measured as velocity in any direction: fitness convergence characteristics of ten independent runs of GP with LMS (a); canonical GP (b); probability of success (c); and snapshots of sample evolved via GP with LMS best-of-run sidewinding Snakebots (d), (e), (f), and (g). The dark trailing circles in (d), (e), (f), and (g) depict the trajectory of Snakebot's center of mass.

ten independent runs (Fig. 6.7a and b) illustrate, GP with LMS is almost twice as fast as canonical GP. Compared to forward locomotion in an unconstrained environment (Fig. 6.6), the velocity in this experiment is superior, and even comparable to the velocity of sidewinding (Fig. 6.5). This seemingly anomalous phenomenon demonstrates a case of emergent intelligence—i.e., an ability of evolution to discover a way to utilize the walls of a "tunnel" as a source of extra grip and as an additional mechanical support for fast yet unbalanced locomotion gaits (e.g., vertical undulation) in an eventually unconstrained environment.

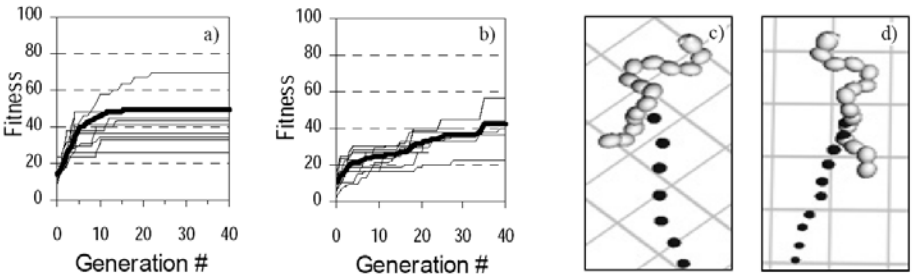


Fig. 6.6. Evolution of locomotion gaits for cases in which fitness is measured as velocity in the forward direction only. Fitness convergence characteristics of ten independent runs of GP with LMS (a), canonical GP (b), and snapshots of sample evolved via GP with LMS best-of-run forward locomotion (c and d).

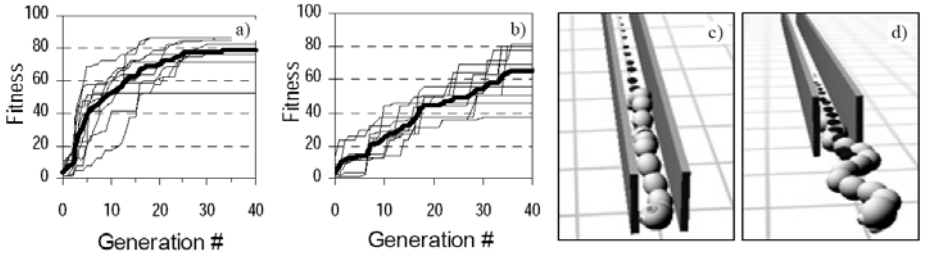


Fig. 6.7. Evolution of locomotion gaits of the Snakebot confined in a narrow "tunnel": fitness convergence characteristics of ten independent runs of GP with LMS (a), canonical GP (b), and snapshots of sample evolved best-of-run gaits at the intermediate (c), and final stages of the trial (d).

6.4.2 Adaptation to Unanticipated Challenging Terrain. Generality of Adapted Gaits

Adaptation in nature is viewed as an ability of species to discover the best phenotypic (i.e., pertaining to biochemistry, morphology, physiology, and behavior) traits for survival in a continuously changing fitness landscape. The adaptive phenotypic traits are the result of beneficial genetic changes that occurred during the course of evolution (phylogenesis) and/or phenotypic plasticity (ontogenesis—learning, polymorphism, polyphenism, immune response, adaptive metabolism, etc.) occurring during the lifetime of the individuals. In our approach we employ GP with LMS for the adaptation of the Snakebot to changes in the fitness landscape caused by a challenging environment and partial damage to one, two, four and eight (out of 15) morphological segments. In all of the cases of adaptation, GP is initialized with a population comprising 20 best-of-run genetic programs, obtained from ten independent runs of evolution of the Snakebot in an unconstrained environment, plus an additional 180 randomly created individuals.

The challenging environment is modeled by the introduction of immobile obstacles comprising 40 small, randomly scattered boxes, a wall with height equal to the

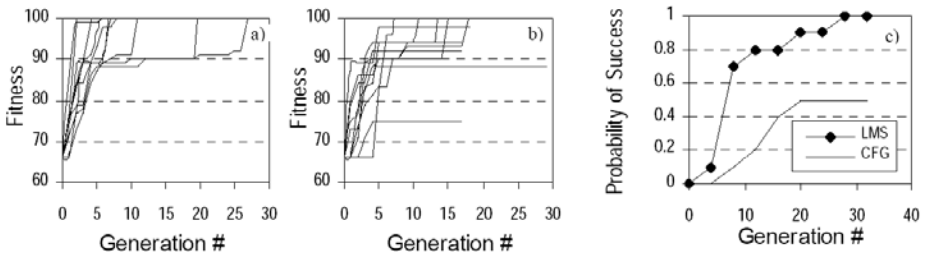


Fig. 6.8. Adaptation of sidewinding locomotion to a challenging environment: fitness convergence characteristics of ten independent runs of GP with LMS (a), canonical GP (b), and probability of success (c).

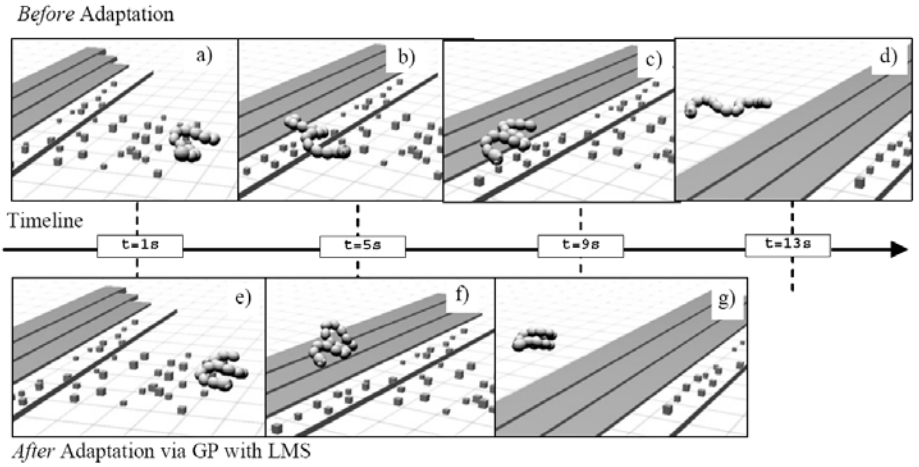


Fig. 6.9. Snapshots illustrating the sidewinding Snakebot, initially evolved in an unconstrained environment, before the adaptation: initial (a), intermediate (b and c), final stages of the trial (d), and after the adaptation to a challenging environment via GP with LMS: initial (e), intermediate (f), and final stages of the trial (g).

0.5 diameters of the cross section of the Snakebot, and a flight of three stairs, each with a height equal to 0.33 diameters of the cross section of the Snakebot. The results of adaptation of the Snakebot, shown in Fig. 6.8, demonstrate that the computational effort (required to reach fitness values of 100 with probability of success 0.9) of GP with LMS is about 20 generations. Conversely, only half of all the runs of canonical GP achieve the targeted fitness value, implying that the corresponding probability of success converges to a value of 0.5. Snapshots illustrating the performance of a Snakebot initially evolved in an unconstrained environment before and after the adaptation (via GP with LMS) to a challenging environment are shown in Fig. 6.9.

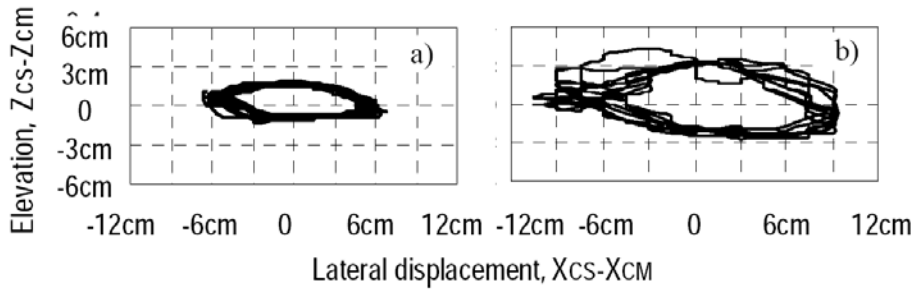


Fig. 6.10. Trajectory of the central segment (cs) around the center of mass (cm) of Snakebot for sample best-of-run sidewinding locomotion before (a) and after the adaptation (b) to a challenging environment.

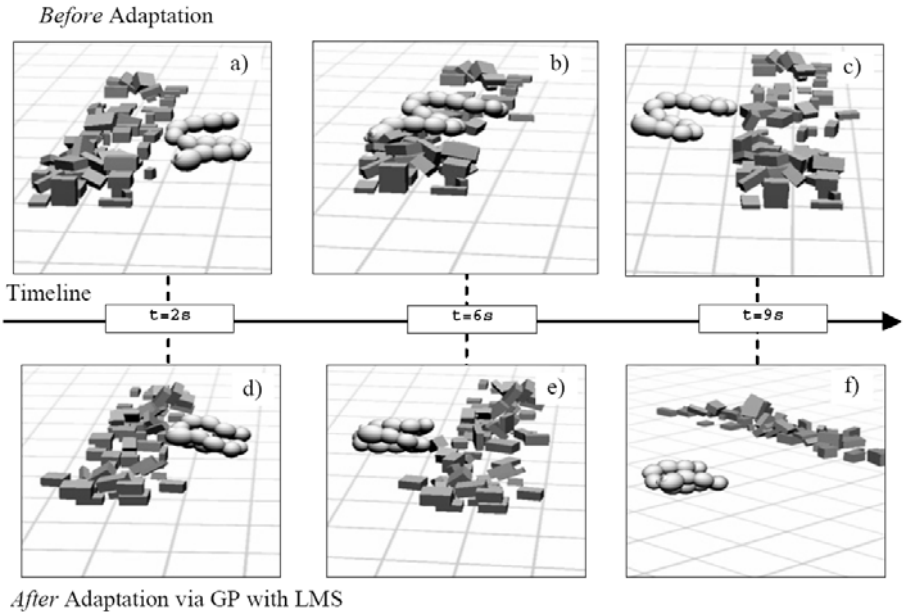


Fig. 6.11. Snapshots illustrating the generality of the sidewinding Snakebot adapted to the known challenging environment as depicted in Fig. 6.9. Before the adaptation the Snakebot overcomes an unanticipated pile of boxes more slowly (a, b and c) than after the adaptation (d, e, and f) via GP with LMS.

The additional elevation of the body required to negotiate the obstacles faster represents emergent know-how in the adapting Snakebot. As Fig. 6.10 illustrates, the trajectory of the central segment around the center of mass of the sample adapted Snakebot (Fig. 6.10b) is twice as high as before the adaptation (Fig. 6.10a).

The generality of the robust sidewinding gaits evolved via GP with LMS is demonstrated by the ease with which the Snakebot evolved in known challenging terrain overcomes various types of unanticipated obstacles, such as a pile of boxes, burial under boxes, and small walls, as illustrated in Figs. 6.11, 6.12, and 6.13.

6.4.3 Adaptation to Partial Damage

The adaptation of the sidewinding Snakebot to partial damage to one, two, four, and eight (out of 15) segments by gradually improving its velocity is shown in Fig. 6.14. Demonstrated results are averaged over ten independent runs for each case of partial damage to one, two, four, and eight segments. The damaged segments are evenly distributed along the body of the Snakebot. Damage inflicted to a particular segment implies a complete loss of functionality of both horizontal and vertical actuators of the corresponding joint.

As Fig. 6.14 depicts, the Snakebot recovers completely from the damage to a single segment, attaining its previous velocity in 25 generations with canonical GP

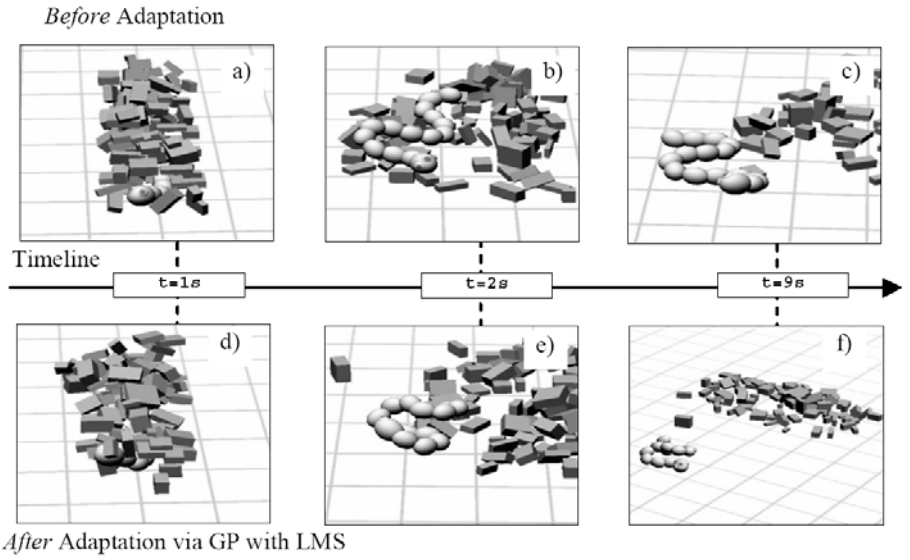


Fig. 6.12. Snapshots illustrating the generality of the sidewinding Snakebot adapted to the known challenging environment as depicted in Fig. 6.9. Before the adaptation the Snakebot emerges from an unanticipated burial under a pile of boxes more slowly (a, b, and c) than after the adaptation (d, e, and f) via GP with LMS.

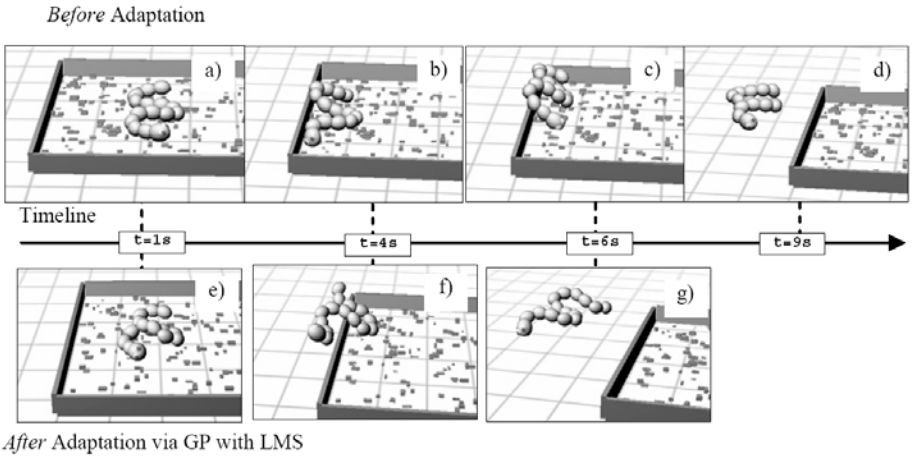


Fig. 6.13. Snapshots illustrating the generality of the sidewinding Snakebot adapted to the known challenging environment as depicted in Fig. 6.9. Before the adaptation the Snakebot clears unanticipated walls forming a pen more slowly (a, b, c, and d) than after the adaptation (e, f, and g). The walls are twice as high as in the known challenging terrain, with their height being equal to the diameter of the cross section of the Snakebot.

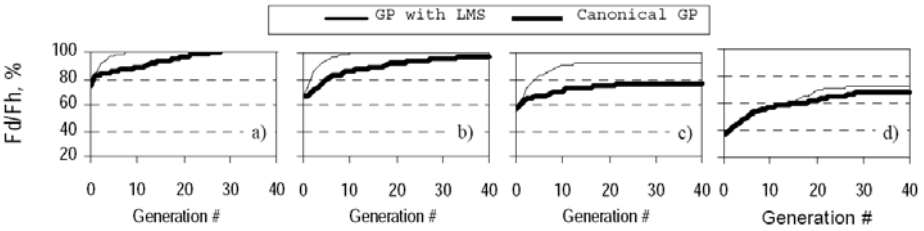


Fig. 6.14. Adaptation of the Snakebot to damage to one (a), two (b), four (c), and eight (d) segments. Fd is the best fitness in the evolving population of damaged Snakebots, and Fh is the best fitness of 20 best-of-run healthy sidewinding Snakebots.

and in only seven generations with GP with LPCSG, resulting in a mean real time of adaptation of a few hours of runtime on a PC featuring an Intel® 3GHz Pentium® four microprocessor and 2GB RAM under Microsoft Windows XP OS. The Snakebot recovers an average of 94% (canonical GP) and 100% (GP with LMS) of its previous velocity in the case in which two segments are damaged. With four and eight damaged segments the degree of recovery is 77% (canonical GP) and 92% (GP with LMS), and 68% (canonical GP) and 72% (GP with LMS), respectively. In all of the cases considered, incorporating LMS contributes to faster adaptation, and in all cases the Snakebot recovers to higher values of velocity of locomotion. The snapshots of the sidewinding Snakebot immediately after damage and after having recovered from the damage of one, two, four, and eight segments are shown in Fig. 6.15. The views of the recovered Snakebot (Fig. 6.15b, d, f, and h) reveal the emergent tendency of increasing the winding angle of locomotion. Moreover, the frontal view of the Snakebot before (Fig. 6.16a)

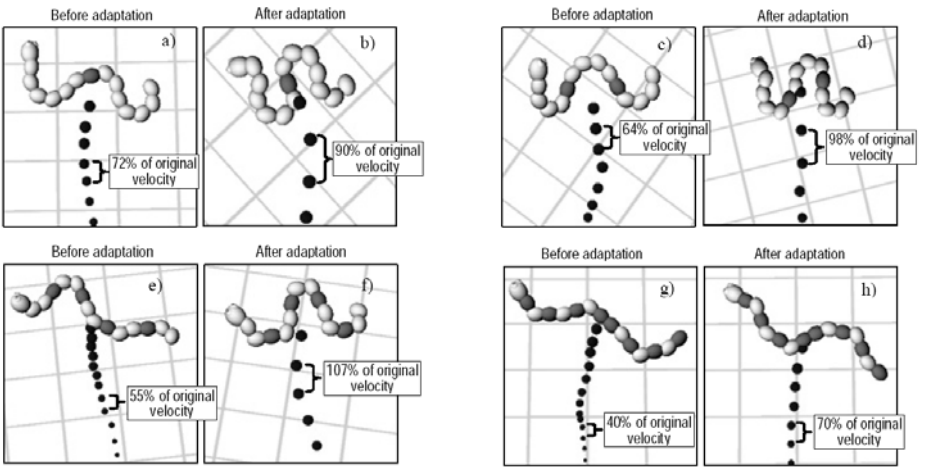


Fig. 6.15. Snapshots of the sidewinding Snakebot immediately after damage to one (a), two (c), four (e), and eight (g) segments and after having recovered from the damage (b, d, f, and h) by adaptation via GP with LPCSG. The damaged segments are shown in a darker shade.

Fig. 6.16. The frontal view of the Snakebot before (a) and after the adaptation (b) to the damage of a single segment. The corresponding views from above of the sidewinding Snakebot are shown in Fig. 6.15a and b, respectively.

and after the adaptation (Fig. 6.16b) to the damage of single segment demonstrates the additional elevation of the adapted Snakebot in a way that is analogous to the adaptation to the challenging environment as illustrated in Fig. 6.10.

6.5 Discussion

The efficiency of incorporating LS in GP depends on several factors, such as the adequacy of the genetic representation of the solution, the size of the search space, and the characteristics of the fitness landscape. Considering the latter issue, we believe that the gradients toward the global optimums are a relevant prerequisite for an efficient evolution. These nondeceptive fitness gradients seemed to appear in the tasks of evolving and adapting the Snakebot as elaborated in the preceding sections. However, in some cases (as illustrated in Figs. 6.9, 6.11, and 6.12) the artifact might be temporarily trapped by obstacles in the challenging environment. Consequently, the eventual evolutionary modifications to the artifact is locomotion patterns might temporarily yield a negligible velocity of locomotion and, consequently, negligibly small fitness values, providing virtually no insight into evolution about the promising areas in the explored search space. The corresponding fitness landscape would feature wide areas covered by low plateaus, which might render simulated evolution to a poorly guided or even a random search with relatively low computational efficiency. The information-driven evolutionary design, which introduces spatiotemporal measures of coordination of the modules that indirectly approximate the fitness function, promises to be an interesting way to address such a problem (Prokopenko et al. 2006).

An alternative approach to addressing the issue of evolving an initially trapped Snakebot is to employ a derivation of the GP with LMS discussed above, in which the probabilities of applying the production rules are learned interactively from the parsed syntax of the Snakebots that, according to the human observer, are believed to exhibit behavioral features that are relevant for overcoming the obstacles (e.g., symmetrical shape, well-synchronized motion of segments, body elevation, etc.). Because these features might not necessarily be exhibited by the current best-performing Snakebots, they would provide the evolution with additional insight about the promising areas in the fitness landscape. The preliminary results indicate that employing such an interactive feature-oriented GP via LMS is associated with improved efficiency in that the locomotion gaits of the Snakebot evolve faster and to higher velocities than those of the canonical GP (Tanev 2006).

6.6 Conclusion

In this work we proposed an approach that incorporates LMS implemented via LPCSG in GP and verified it in terms of the efficiency of evolution and adaptation of locomotion gaits of a simulated Snakebot. We introduced a biased mutation in which the probabilities of applying different production rules with multiple right-hand-side alternatives in the LPCSG depend on the context, with these probabilities being "learned" from the aggregated reward values obtained from the evolved best-of-generation Snakebots. Empirically obtained results confirmed that employing LMS contributes to the improvement of computational effort of both the evolution of Snakebot's fastest possible locomotion gaits for various fitness conditions and the adaptation of these locomotion gaits to a challenging environment and degraded mechanical abilities.

Recent discoveries in molecular biology and genetics suggest that mutations do not happen randomly in nature (Caporale 2003; Kirschner and Gerhart 2005) Instead, some fragments of DNA tend to repel the mutations, whereas other fragments seem to attract them. It is assumed that the former are related to the very basics of life, and therefore any mutation within them might be potentially fatal to the species. We consider the ability of the Snakebot to move as an analogy of these very basics of life. Preserving the corresponding genotypic areas from mutations and focusing on genetic changes that facilitate the discovery of the beneficial phenotypic properties (e.g., additional elevation of the body and increased winding angle) of the already evolved fast locomotion gaits improves the efficiency of evolution and adaptation of the Snakebot to challenging environments and partial damage. The proposed approach contains no domain specifics and therefore can be incorporated into genetic programming for solving a wide range of problems from various problem domains.

Considering the situational awareness as a necessary condition for any intelligent autonomous artifact, in future work we would like to investigate the feasibility of incorporating sensors that allow the Snakebot to explicitly perceive the surrounding environment. We are especially interested in sensors that do not compromise the robustness characteristics of the Snakebot—such as, for example, Golgi's tendon receptors incorporated inside a potentially completely sealed Snakebot.

Acknowledgments

The author thanks Katsunori Shimohara, Thomas Ray, and Andrzej Buller for their support of this work.

References

Andrusenko Y. (2001). *Russian Culture Navigator: Miturich-Khlebnikovs: Art Trade Runs in the Family*, available at: http://www.vor.ru/culture/cultarch191_eng.html.

Angeline, P. J. (1994). Genetic programming and emergent intelligence. In Kinnear, K.E. Jr., editor, *Advances in Genetic Programming*, pages 75–98, MIT Press, Cambridge, MA.

Antonisse, J. (1991). A grammar-based genetic algorithm. In G.J.E. Rawlins, editor, *Foundations of the Genetic Algorithm Workshop (FOGA)*, pages 193–204. Bloomington USA Morgan Kaufmann, CA.

Bongard, J.C., and Lipson H. (2004). Automated damage diagnosis and recovery for remote robotics. In *IEEE Proceedings of the 2004 International Conference on Robotics and Automation (ICRA 2004)*, pages 3545–3550. IEEE, New York.

Bosman, P., and de Jong, E. (2004). Learning probabilistic tree grammars for genetic programming. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature PPSN-04*, pages 192–201. Birmingham, UK; Springer.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E. (2000). Extensible Markup Language (XML) 1.0, Second Edition, W3C Recommendation, available at http://www.w3.org/TR/REC-xml/

Caporale, L. H. (2003). *Darwin in the Genome: Molecular Strategies in Biological Evolution*. McGraw-Hill/Contemporary Books, New York.

Dowling, K. (1997). *Limbless Locomotion: Learning to Crawl with a Snake Robot*. Doctoral dissertation, Technical Report CMU-RI-TR-97-48. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addision-Wesley,

Hirose, S. (1993). *Biologically Inspired Robots: Snake-like Locomotors and Manipulators*. Oxford University Press. Oxford.

Kamio, S., Mitsuhashi, H., and Iba, H. (2003). Integration of genetic programming and reinforcement learning for real robots. In E. Cantú-Paz, J. A. Foster, K. Deb, L. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. K. Standish, G. Kendall, S. W. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, J. F. Miller, editors, *Proceedings of the Genetic and Evolutionary Computations Conference (GECCO 2003)*, pages 470–482, Springer, Berlin.

Kimura, H., Yamashita, T., and Kobayashi, S. (2001). Reinforcement learning of walking behavior for a four-legged robot. In *Proceedings of 40th IEEE Conference on Decision and Control, Orlando, FL*, pages 411–416.

Kirschner, M.W. and Gerhart, J.C. (2005). *The Plausibility of Life: Resolving Darwin's Dilemma*. Yale University Press, New Haven, CT.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.

Ito, K., Kamegawa, K. and Matsuno, F. (2003). Extended QDSEGA for controlling real robots—acquisition of locomotion patterns for snake-like robot. In *IEEE Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2003)*, pages 791–796. IEEE, New York.

Levitan, I. B., and Kaczmarek, L. K. (2002). *The Neuron: Cell and Molecular Biology*. Oxford University Press, New York

Mahdavi, S., and Bentley, P.J. (2003). Evolving motion of robots with muscles. In *Proceedings of EvoROB2003, the 2nd European Workshop on Evolutionary Robotic (EuroGP 2003)*, pages 655–664. Essex, UK.

O'Neill, M. and Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*, Kluwer, Norwell, USA.

Pelikan M., Goldberg D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, Orlando, USA pages 525–532.

Prokopenko, M., Gerasimov, V., and Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J. C. T., Marocco, D., Meyer, J.-A., Miglino, O., and Parisi, D., editors, *From Animals to Animats 9: 9th International Conference on the Simulation of Adaptive Behavior (SAB 2006), Rome, Italy, September 25–29 2006*, volume 4095 of *Lecture Notes in Computer Science*, pages 558–569. Springer.

Salustowicz, R. and Schmidhuber, J. (1997). Probabilistic incremental program evolution. *Evolutionary Computation*, MIT Press, 5(2):123–141.

Shan Y., McKay, R.I, and Baxter R. (2004). Grammar model-based program evolution. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation, 20-23 June, Portland, Oregon*, pages 478–485.

Smith, R. (2006). *Open Dynamics Engine*, available at http://q12.org/od

Takamura, S., Hornby, G. S., Yamamoto, T., Yokono, J. and Fujita, M. (2000). Evolution of dynamic gaits for a robot. In *Proceedings of the IEEE International Conference on Consumer Electronics*, pages 192–193, Los Angeles.

Tanev, I., and Ray, T. (2005). Evolution of sidewinding locomotion of simulated limbless, wheelless robots. *Artificial Life and Robotics*, 9:117–122, Springer, Tokyo.

Tanev, I., Ray, T., and Buller, A. (2005). Automated evolutionary design, robustness and adaptation of sidewinding locomotion of simulated snake-like robot, *IEEE Transactions on Robotics*, 21:632–645.

Tanev, I. (2006). Interactive learning of mutation strategies in genetic programming. In *Proceedings of the 5th Joint Symposium between Chonnam National University and Doshisha University, Chonnam University, Kwangju, Korea*, pages 83–87. Chonnam Univ. Press.

Wong, M. L. (2005). Evolving recursive programs by using adaptive grammar based genetic programming, *Genetic Programming and Evolvable Machines*, 6:421–455.

7

Self-Organization as Phase Transition in Decentralized Groups of Robots: A Study Based on Boltzmann Entropy

Gianluca Baldassarre

7.1 Introduction

An important goal of collective robotics (Dudek et al. 1996; Cao et al. 1997; Dorigo and Sahin 2004; Dorigo et al. 2004) is the development of multirobot systems capable of accomplishing collective tasks without centralized coordination (Kube and Zhang 1993; Holland and Melhuish 1999; Ijspeert et al. 2001; Quinn et al. 2003). From an engineering point of view, decentralized multirobot systems have several advantages vs. centralized ones, at least for some tasks. For example, they are more robust with respect to the failure of some of their component robots, do not require a control system or robot with sophisticated computational capabilities to manage the centralized control (Kube and Bonabeau 2000), have a high scalability with respect to the whole system's size (Baldassarre et al. 2006, 2007a), and tend to require simpler robots, as due to the more limited need for communication, they can often rely upon implicit coordination (Beckers et al. 1994; Trianni et al. 2006).

Decentralized coordination is usually based on self-organizing principles. Very often research on decentralized multirobot systems makes a general claim on the presence of these principles behind the success of the studied systems, but it does not conduct a detailed analysis of which specific principles are at work, nor does it attempt to measure their effects in terms of the evolution of the system's organization in time or to analyze the robustness of its operation vs. noise (e.g., see Holland and Melhuish 1999; Krieger et al. 2000; Kube and Bonabeau 2000; Quinn et al. 2003). This chapter studies some of these issues in a multirobot system that was presented in detail elsewhere (Baldassarre et al. 2003, 2006, 2007, 2007a). This system is formed by robots that are physically connected and have to coordinate their direction of motion to explore an open arena without relying on centralized coordination. The robots are controlled by an identical neural network whose weights are evolved through a genetic algorithm. Through this algorithm the system develops the capacity to solve the task on the basis of self-organizing principles. The goal of this chapter is to present some preliminary results that show how such principles lead the organization of the system, measured through a suitable index based on Boltzmann entropy, to arise in quite an abrupt way if the noise/signal ratio of the signal that allows the robots to coordinate is

slowly decreased. In this regard, the chapter argues, on the basis of theoretical arguments and experimental evidence, that such sudden emergence of organization shares some properties with the phase transitions exhibited by some physical systems studied in physics (Anderson 1997).

The rest of the chapter is organized as follows. Section 7.2 presents a qualitative description of the mechanisms that are usually behind self-organization and an index, based on Boltzmann entropy, that can be used to measure the *synchronic* level of the order of a system composed of many dynamical parts. Section 7.3 illustrates the robots forming the multirobot system considered here, the collective task tackled with it, the neural controller of the robots, and the genetic algorithm used to evolve it. Section 7.4 analyzes the behavior of the single robots developed by the genetic algorithm, and the effects it has at the collective level. Section 7.5 uses the entropy index to show that when the noise/signal ratio related to the signal used by the robots to coordinate is slowly decreased, the level of order of the robotic system behaves as some global organization parameters observed in phase transitions of some physical systems. Finally, Section 7.6 draws the conclusions.

7.2 Mechanisms of Self-Organization, Phase Transitions, and Indexes to Measure the Organization Level of Collective Systems

Prokopenko et al. (2007) (see also Chapter 1 in this volume) suggest that self-organization is characterized by three features: (a) it causes the parts forming a collective system to acquire global coordination; (b) this coordination is caused by the local interactions and information exchange between the parts composing the system and not by a centralized ordering mechanism; (c) the system passes from less to more organized states. This section first tackles points (a) and (b) from a qualitative perspective by presenting three basic mechanisms that usually underlie self-organization. It then presents an index based on Boltzmann entropy that can be used to measure the level of order of a collective system at a given instant of time. This index can be used, as illustrated in the succeeding sections, to measure the level of organization of a multirobot system under the action of self-organizing processes and hence to study point (c). Finally the section presents some theoretical arguments in favor of the hypothesis that in some cases the dynamics of order exhibited by self-organizing multirobot systems might have the features of phase transitions studied in physics. These arguments are supported by the preliminary experimental results presented in Section 7.5.

7.2.1 Qualitative Mechanisms of Self-Organization

Self-organizing processes involve systems composed of several, generally similar components, and usually (always?) rely on three basic principles (Camazine et al. 2001): (a) random fluctuations; (b) positive feedback; (c) negative feedback, which will now be illustrated in detail.

The elements composing self-organizing systems are usually dynamic in the sense that they can assume one state among a certain number of possible states at each time

step and pass from state to state in time. Fully disorganized systems are those in which each component passes from state to state in a random fashion. A typical feature of such systems is that the distribution of the components over the possible states tends to be uniform, i.e., *symmetric* (e.g., a group of fish swimming randomly in an aquarium have a quite uniform distribution in the water).

The symmetry of a collective system formed by components exhibiting a random dynamics tends to be imperfect in the sense that it tends to have *random fluctuations* in time due to noise (e.g., there are some zones of the aquarium with a slightly higher density of fish). Now consider the possibility that each component of the system does not move (only) randomly, but tends to assume the states assumed by some other components of the system, i.e., it individually follows a *conformist rule* of the kind, "I do what you do" (e.g., fish move to portions of space where other fish are, so as to minimize the chance of being found alone by predators). In this condition, it might happen that some *random fluctuations are amplified*: in fact the more components that assume a certain state vs. other states, the more components among the remaining ones that will tend to imitate their state, so causing an exponential avalanche effect with a consequent *symmetry break* of the initial uniform distribution (e.g., the fish tend to cluster and form a whole school). The process that leads to this amplification is called *positive feedback*. In all real systems, the action of positive feedback tends to be counterbalanced by *negative feedback.* The latter might assume the form of an active process (e.g., the fish tend to cluster to avoid predators, but they also tend to keep at a certain minimal distance to avoid collisions) or a passive process (e.g., all fish have converged to the same zone in space) so the process of convergence stops. Starting from an initial uniform distribution, and after a first exponential convergence of the elements of the system to similar states due to positive feedback, negative feedback will start to slow down the convergence process. In this respect, negative feedback tends to operate with a strength positively related to the number of elements that have already converged to the same states (e.g., to avoid collisions the fish's "repulsion" behavior might be implemented with more vigor in space areas with higher densities of conspecifics as such densities correspond to smaller distances and higher chances of collision). For this reason negative feedback usually increases to levels that fully counterbalance the effect of positive feedback. At this point the system's overall state usually tends to reach equilibrium (e.g., the fish school's density remains within a certain range; for examples of simulations of flocks, herds, and schools of animals, see the seminal paper of Reynolds (1987) and the literature that followed it linked in the web page http://www.red3d.com/cwr/boids/).

7.2.2 An Index to Measure the Synchronous Level of Organization of Collective Systems Based on Boltzmann Entropy

The index used to measure the level of order of the group of robots studied here is based on Boltzmann entropy. Note that the index can be used to measure the level of organization of a collective system independently of the fact that such organization is the result of the action of self-organizing or of centralized coordination mechanisms. Boltzmann entropy has been proposed in mechanical statistics to measure the level of

disorder that characterizes a system formed by a set of N gas molecules that occupy a given portion of space. This portion of space is divided into an arbitrary number of cells, C, each having a constant volume. (In general the number of cells will influence the outcome of the application of the index, but, as we will see, the index can be suitably normalized to avoid this problem.) The index is based on the assumption that the elements composing the system move randomly. This implies that at any time step an element can occupy any cell with a constant probability $1/C$ (the cell occupied by the element will constitute its *state*). To give an example of this, consider the case of the robotic system studied here. This system is composed of $N = 40$ robots. Each robot can assume a given direction of motion ranging over a 1D closed space that ranges over $[0°, 360°]$ degrees. If this space is divided into $C = 8$ cells of constant size, at each time step the probability that an element occupies a given cell is equal to $1/8$.

The computation of the index levels is based on the so called microstates and macrostates of the system. A *microstate* corresponds to the set of individual states of the elements in a given time step. For example, in a system with $N = 2$ and $C = 2$, the microstate is the vector (c_1, c_2), where c_n is the cell occupied by the element n. Note that the microstate is a vector and not a set, i.e., the order of the c_n states of the elements is relevant: this is a consequence of the fact that the *identity of the elements is assumed to be distinguishable*. So, e.g., given a system with $N = 2$ and $C = 2$, the microstate where the first element occupies the first cell and the second element occupies the second cell is different from the microstate where the first element occupies the second cell and the second element occupies the first cell, even if in both cases the system has one element in the first cell and one element in the second. As each element can be in one of C possible different states, the number of different possible microstates is C^N.

With N_i indicating the number of elements in cell i, a *macrostate* of the system is defined as the *distribution* $(N_1, N_2, \ldots, N_i, \ldots, N_C)$ of the elements over the cells, *without considering the identity* of the elements. An example of distribution for the system with $N = 2$ and $C = 2$ is (0, 2), meaning that there are zero elements in the first cell and two elements in the second. Each macrostate is (usually) composed of several possible microstates as the distribution of elements over the cells that correspond to it can be obtained in different ways. For example, in the $N = 2, C = 2$ system, the macrostate (1, 1) with one element in each cell is composed of two microstates, i.e., (1, 2) and (2, 1). The other two macrostates (2, 0) and (0, 2), respectively, with both elements in the first and the second cell, are each composed of only one microstate, respectively, (1, 1) and (2, 2).

Boltzmann entropy E_m refers to the macrostate m of the system at a given time step and is defined as follows:

$$E_m = k \ \ln[w_m], \tag{7.1}$$

where w_m is the number of microstates of m, $\ln[\cdot]$ is the natural logarithm, and k is a scaling constant.

As at any time step the probability of having any microstate is constant and equal to $1/C^N$; the probability that the system is in a given macrostate is proportional to the number of microstates that compose it (this probability is equal to w_m/C^N). Now

consider the possibility that an *ordering mechanism* (e.g., a flow of energy that goes through the system) starts to operate on the elements of the system previously subject only to noise. This mechanism is "ordering" in the sense that it drives the system toward macrostates composed of few microstates, so it operates *against the noise*, i.e., against the evolution that the system would undergo if driven only by noise. The important point for Boltzmann entropy is that as the elements of the system wander across the different states *due to noise*, and hence the system wanders across the different corresponding microstates, at a given time step the system has a *high probability* of being in macrostates that are formed by many microstates vs. macrostates that are formed by few microstates. As Boltzmann entropy is positively related with the number of microstates that compose the macrostate of the system, it can be considered a *measure of the disorder of the system* caused by the random forces acting on its composing elements and operating against the ordering mechanisms eventually existing within it. This also implies that Boltzmann entropy can be used as an index to detect the *presence* and *level* of effectiveness of ordering mechanisms operating in the system: the lower the value of the index, the stronger the effectiveness of such mechanisms.

Note that highly disordered macrostates correspond to situations in which the elements of the system tend to be more equally distributed over the cells (i.e., macrostates composed of many microstates), hence to situations where the system is highly *symmetric*. On the other hand, ordered macrostates correspond to situations where the system is more *asymmetric*, e.g., macrostates where the system's elements gather in a few cells (i.e., macrostates composed of relatively few microstates). In this respect, ordering mechanisms operating on the system tend to lead it from symmetric to more asymmetric global states.

The reader should note an important feature of the index of disorder used here: it allows computation of the level of disorder of a dynamical system *at a given time step*, whereas many other indexes applied to dynamical systems, such as the entropy rate and the excess entropy, are used to capture the regularities of the states visited by the systems in time (Feldman 1998; Prokopenko et al. 2006). This property allows for using the index to study how the level of order of systems evolves in time, as done here and in Baldassarre et al. (2007). Intuitively, the reason the index can compute the level of disorder of a system at an instant of time, i.e., on the basis of a "synchronic picture" of it, is that unlike other indexes it does not need to compare the states that the system assumes in time in order to estimate the probabilities of such states. But it rather computes such probabilities on the basis of the *potential microstates* that the system *might* have assumed if driven by sheer random forces.

Calculating the specific value of the index for a particular macrostate m assumed by a system requires computing the number w_m of microstates that compose it. This number can be obtained as follows:

$$w_m = \frac{N!}{N_1!\, N_2!\, ...\, N_C!} \qquad \sum_{i=1}^{C} N_i = N, \tag{7.2}$$

where N_i is the number of elements in the cell c, and "!" is the factorial operator. The formula relies on the fact that there are $((N)(N-1)\ldots(N-N_1+1))/N_1!$

different possible sets of elements that can occupy the first cell, $((N - N_1)(N - N_1 - 1) \ldots (N - N_1 - N_2 + 1))/N_2!$ different sets of elements that can occupy the second cell for each set of elements occupying the first cell, and so on. The expression for w_m is given by the multiplication of these elements referring to all the C cells. Substituting Eq. (7.2) into the Eq. (7.1) of the index, one has.

$$E_m = k \ \ln[w_m] = k \ \ln\left[\frac{N!}{N_1!\ N_2!\ \ldots\ N_C!}\right] = \ k\left(\ln[N!] - \sum_{i=1}^{C} \ln[N_i!]\right). \quad (7.3)$$

Once N and C are given, the maximum entropy is equal to the entropy of the macrostate, where the N elements are equally distributed over the cells. This allows setting k to one divided by the maximum entropy, obtaining, from Eq. (7.3), a normalized entropy index ranging in $[0, 1]$:

$$\begin{aligned} E_m = k \ \ln[w_m] &= \frac{1}{\ln\left[\dfrac{N!}{((N/C)!)^C}\right]} \ \ln[w_m] \\ &= \frac{1}{\ln[N!] - C \ \ln[(N/C)!]}\left(\ln[N!] - \sum_{i=1}^{C} \ln[N_i!]\right). \end{aligned} \quad (7.4)$$

Last, the calculation of the index can avoid the computation of the factorials, which becomes infeasible for increasing integers, by using the Stirling approximation:

$$\ln[n!] \approx \left(n + \frac{1}{2}\right) \ln[n] - n + \ln\left[\sqrt{2\pi}\right]. \quad (7.5)$$

Stirling's approximation gives increasingly good results for integers n of increasing size (e.g., the error of approximation is less than 0.5% for $n > 20$).

7.2.3 An Hypothesis: Self-Organization of Multirobot Systems as a Phase Transition

One of the main contributions of this chapter is to present some preliminary results that hint at the fact that the self-organization of robotic systems such as those considered here might have the features of phase transitions such as those studied in physics. According to Wikipidia (http://en.wikipedia.org/wiki/Phase_transition), a phase transition can be defined as follows: "In physics, a phase transition, or phase change, is the transformation of a thermodynamic system from one phase to another. The distinguishing characteristic of a phase transition is an *abrupt sudden change* in one or more *physical properties*, in particular the heat capacity, with a *small change* in a thermodynamic *variable* such as the temperature" (italics added). The distinguishing feature of a phase transition is hence the fast change of a variable related to the collective level of a system (e.g., the heat capacity of a gas, which is the capacity of a whole gaseous system to absorb energy when temperature changes by a certain amount) when a variable related to the behavior of the composing elements (e.g., the average noisy movement

of the molecules of a gas, captured by the temperature) is slowly changed and passes a *critical value* that characterizes the phase transition.

The diagram in Fig. 7.1 shows an example of a phase transition in a physical system, illustrated through a result obtained in physics with a spin-1 Icing model related to finite spin systems (Tsai and Salinas 1998). This example shows how the magnetization properties of the spin system undergo an abrupt change when the temperature of the system is slowly decreased below a critical value.

Here we suggest that the dynamics of organization generated by self-organizing principles in multirobot systems might share some features with that of the global organization exhibited by some physical systems undergoing a phase transition. The suggestion stems from the following considerations: The behavior of individual robots is affected by noise that influences their sensors' reading and their actuators' performance. This noise causes the robots to act in a random disorganized fashion. On the other hand, the controller of the robots might implement an "ordering mechanism" of the kind "I do what you do," which tends to generate self-organization within the system. However, in order to lead the whole system to successfully self-organize (i.e., all robots converge on the same behavior), the ordering mechanism has to overcome the effects of noise. This requires three conditions: (a) the signal that is perceived by the robots through the sensors that inform them concerning the behavior of the other robots (i.e., it allows the robots to know "what you do") is *sufficiently* high with respect to noise; (b) the commands issued to the motors (i.e., the "I do" part) are *sufficiently* effective and succeed in overcoming the noise affecting the actuators' response; (c) the controller is capable of implementing a "conformist principle" that self-organization needs to function (i.e., to implement the association "what you do → I do").

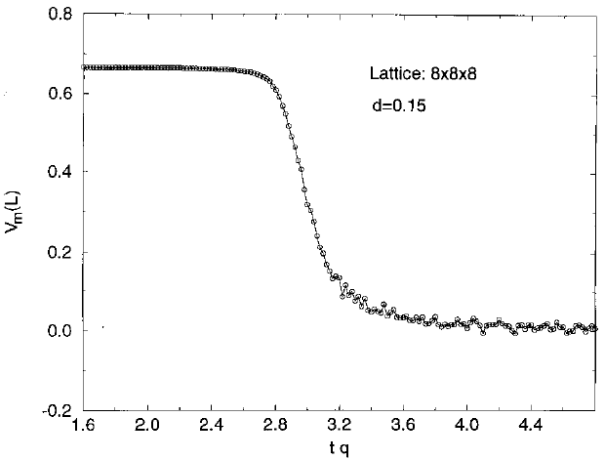


Fig. 7.1. Example of a phase transition studied in physics. y-axis: a measure of magnetization (fourth-order cumulant) in a spin-1 Icing model. x-axis: temperature. Reported from Tsai and Salinas (1998: copyright *Brazilian Journal of Physics*).

These considerations suggest the following prediction: in the case in which the actuators are sufficiently reliable and the controllers are sufficiently effective, if the noise/signal ratio related to the robots' sensors is slowly decreased starting from high values, then the organization of the system generated by self-organizing principles should emerge *abruptly*, as in phase transitions studied in physics. The fact that such order should emerge "abruptly" is due to the fact that once self-organization succeeds in amplifying some random fluctuations vs. noise, overcoming the "noise barrier" that initially prevents the emergence of the system's organization (i.e., that continuously disrupts the asymmetries generated by the random fluctuations), then the positive feedback mechanism should generate a self-reinforcing process that will further strengthen the signal that forces the robots to adopt the same behavior. Consequently, such a signal definitely overcomes noise and the system "remains locked" in the organized phase, resisting external perturbations due to noise. Section 7.5 presents some preliminary results that support this prediction and the related explanation.

7.3 Robots and Task

The scenario used for the experiments consists of a group of simulated robots (from 4 to 36, see Figs. 7.2 and 7.6) set in an open arena. The robots are physically linked (they are manually assembled before the evolution and tests) and have to harmonize their direction of motion in order to move together as far as possible from the initial position in a given amount of time.

The simulation of the robots was carried out with a C++ program based on Vortex^{TM} SDK, a set of commercial libraries that allow programming of realistic simulations of dynamics and collisions of rigid bodies in three dimensions. The simulation of each robot was based on the prototype of a hardware robot that was built within the project SWARM-BOTS funded by the European Union (Mondada et al. 2004; see Fig. 7.2). Each robot was composed of a cylindrical turret with a diameter of 5.8 cm and a chassis with two motorized wheels at the two sides and two caster wheels at the front and at the rear for stability. The simulated robot was half the size of the hardware robot, which decreased the weights of the simulated bodies and so allowed for increasing the simulation step of Vortex and decreasing the computational burden of the simulations (see below).

The chassis was capable of freely rotating with respect to the turret through a further motor. This motor was activated on the basis of the difference of the activation of the motors of the two side wheels to ease the robots' turning while it was physically linked to other robots (see Baldassarre et al. 2006 for details). The turret was provided with a gripper through which the robot could grasp other robots: this gripper was simulated through a rigid joint connecting the robots since our work focused on the behavior of groups of robots that were physically linked during the whole duration of the experiments. The gravitational acceleration coefficient was set at 9.8 cm/s^2 and the maximum torque of the wheels' motors was set at 70 dynes/cm. These low parameter

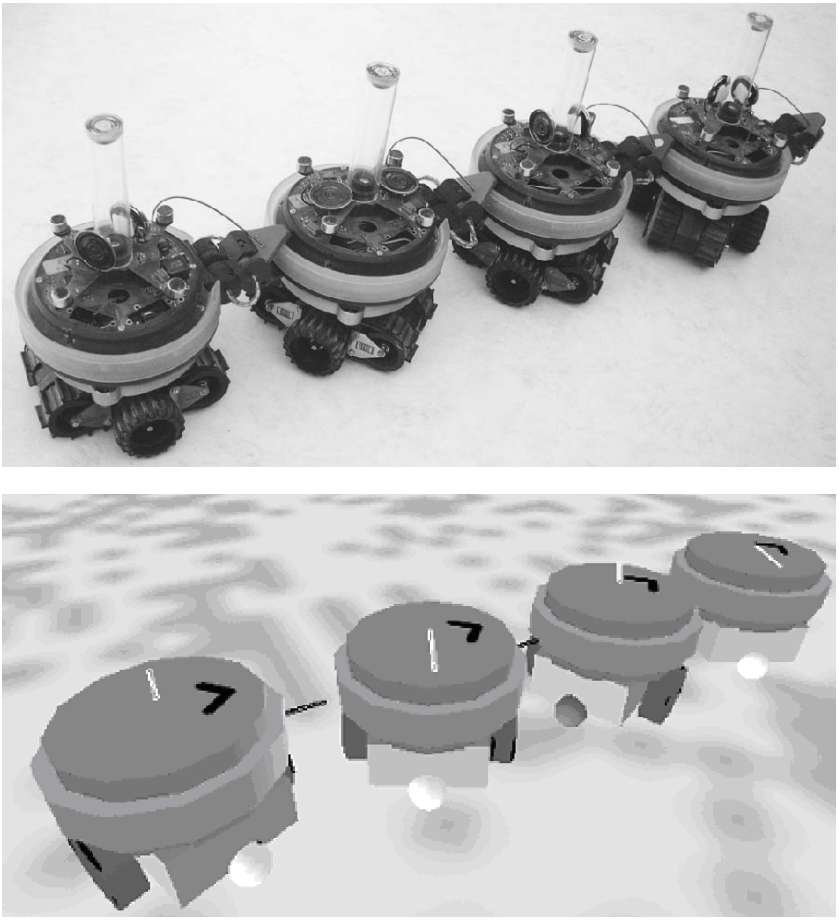


Fig. 7.2. Top: The hardware robots. Bottom: Each simulated robot is made up of a chassis to which two motorized cylindrical wheels and two smaller caster wheels are attached (the visible dark-gray caster wheel marks the front of the chassis). The chassis supports a cylindrical turret (the arrow on the turret indicates its orientation).

settings, together with the small size of the robots, allowed the use of a relatively long integration time step (100 ms) in Vortex. This was desirable since simulations based on Vortex are computationally very heavy. The speed of the wheels was updated by the robots' controllers every 100 ms and could vary within ± 5 rad/s.

Each robot had only a special sensor called a *traction sensor* (introduced for the first time in Baldassarre et al. 2003), which was placed between the turret and the chassis. The sensor indicated to the robot the angle (with respect to the chassis orientation) and the intensity of the force that the turret exerted on the chassis. During the tests this force was caused by the physical interactions among the robots, in particular by

the mismatch of the direction of movement of the robot's chassis with respect to the movement of the robots attached to its turret. Note that if one assumes a perfect rigidity of the physical links, the turrets and the links of the group of robots formed a whole solid body, so the traction measured the mismatch of movement between the robot's chassis and the rest of the group. Traction, seen as a vector, was affected by a 2D noise of $\pm 5\%$ of its maximum length.

The controller of each robot was a two-layer feed-forward neural network. The input layer was composed of four sensory units that encoded the traction force from four different preferential orientations with respect to the chassis's orientation (rear, left, front, and right). When the angle was within $\pm 90^\circ$, each of these units had an activation proportional to the cosine of the angle between the unit's preferential orientation and the traction direction. With angles other than $\pm 90^\circ$, the units had a zero activation. The units' activation was also multiplied by the intensity of the traction normalized in $[0, 1]$. The last unit of the input layer was a bias unit that was constantly activated with 1. The two sigmoid output units were used to activate the wheels' motors by mapping their activation onto the range of the desired speed motor commands that varied ± 5 rad/s.

The connection weights of the neural controllers were evolved through an evolutionary algorithm (Nolfi and Floreano 2001). Initially the algorithm created a population of 100 random genotypes, each containing a binary encoding of the ten connection weights of the neural controller (the weights ranged over ± 10). The neural controller encoded by a genotype was duplicated for a number of times equal to the number of robots forming a group, and these identical controllers were used to control the robots themselves (so the robots were "clones").

Groups of four robots connected to form a line were used to evolve the controllers. Each group was tested in five epochs, each lasting 150 cycles (15 s). At the beginning of each epoch the robots were assigned random chassis orientations. The 20 genotypes corresponding to the groups with the best performance of each generation were used to generate five copies each. Each bit of these copies was mutated (flipped) with a probability of 0.015. The whole cycle composed of these testing, selecting, and reproducing phases was repeated 100 times (generations). The whole evolutionary process was replicated 30 times by starting with different populations of randomly generated genotypes. Note that in this evolutionary algorithm one genotype corresponds to one robot group, and the groups compete and are selected as wholes (the group is the unit of selection of the genetic algorithm). This allows one to obtain groups composed of highly cooperating individuals, thus avoiding the risk of the emergence of "free rider" individuals within them.

The genetic algorithm selected the best 20 genotypes (groups) of the population of each generation on the basis of a fitness criterion that captured the ability of the groups to move as straight and as fast as possible. In particular, the Euclidean distance covered by each group from the starting point to the arrival point was measured and averaged over the five epochs. To normalize the value of the fitness within $[0, 1]$, the distance averaged over the five epochs was divided by the maximum distance covered by a single robot moving straight at maximum speed in 15 s (one epoch).

7.4 Analysis of the Emerged Self-Organizing Behavior at the Individual and Collective Level

The graph of Fig. 7.3 shows how the fitness of the best group and the average fitness of the whole population of 100 groups increase throughout the generations in one evolutionary run. Testing the best groups of the last generation of each of the 30 evolution replications for 100 epochs showed that the best and the worst group have performances of 0.91 and 0.81 respectively. This means that all the evolutionary runs produce groups that are very good at coordinating and moving together.

In what follows the functioning of the evolved behavior is described briefly at the individual level and then at the collective level, focusing on the controller that emerged in the 30th run of evolution (one with top fitness). Overall, the behavior of single robots can be described as a "conformist behavior": the robots tend to follow the movement of the group as signaled by their traction sensors. Figure 7.4 shows more in detail the commands that the controller issues to the motors of the wheels in accordance with different combinations of intensities and angles of traction. If a robot is moving toward the same direction of motion as the group, it perceives a zero or low traction from the front (around 180°): in which case it keeps moving straight. If the robot is moving in one direction and the group moves toward its left-hand side, it tends to perceive a traction from the left (around 90°) and as a consequence turns left. Similarly, if the robot is moving in one direction and the group moves toward its right-hand side, it tends to perceive a traction from the right (around 270°) and as a consequence turns right. Finally, if the robot moves in the opposite direction with respect to the group's movement, it perceives a traction from the rear (around 0°), in which case it tends to move straight, but since this is an unstable equilibrium state situated between the behaviors of turning left and right, the robot soon escapes owing to noise.

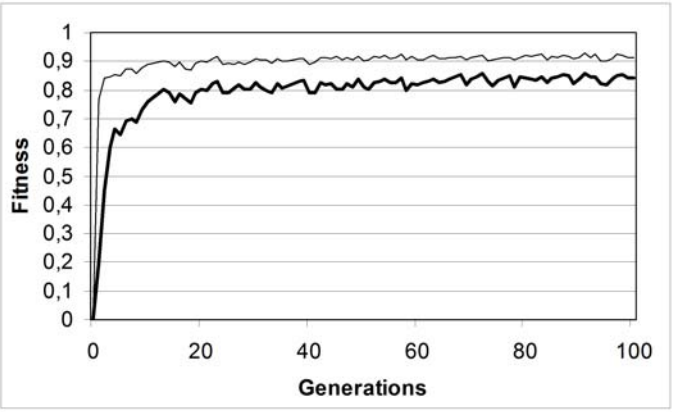


Fig. 7.3. The fitness (y-axis) of the best robot group (thin curve), and average of the whole population (bold curve), across the 100 generations of one of the best evolutionary processes (x-axis).

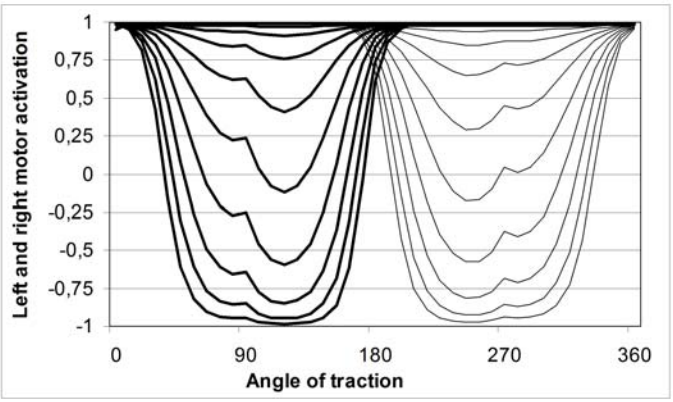


Fig. 7.4. The graph shows how a robot's left motor (bold curves) and right motor (thin curves) react to a traction force with eleven different levels of intensity (different bold and thin lines) and angles measured clockwise from the rear of its chassis (x-axis). The speed of the wheels (y-axis) is scaled between -1 (corresponding to a wheel's maximum backward speed) and $+1$ (wheel's maximum forward speed).

When the evolved robots are tested together, one can observe that they start to pull and push in different directions selected at random. In fact initially there is symmetry in the distribution of the directions of motion over $360°$. Chance causes some robots to move in directions. If one of these random fluctuations eventually gains enough intensity that the other robots feel a traction in that direction, it breaks the initial symmetry: other robots start to follow that bearing, and in so doing they further increase the traction felt by the nonaligned robots toward the same direction. The whole group will hence rapidly converge toward the same direction of motion: the positive feedback mechanism succeeds in amplifying one of the initial random fluctuations, thus causing an avalanche effect that rapidly leads the whole group to coordinate.

It is important to note that the common direction of motion that emerges in one coordinated motion test is the result of a collective decision based on the amplification of some fluctuations that depend on the robots' initial random orientations. As a consequence, as shown in Fig. 7.5, if the test is repeated more times the group's direction of motion that emerges is always different.

Moreover, it is important to note that in some tests in which the robots' chassis have particular initial orientations, the group starts to rotate around its geometrical center. This collective behavior is a stable equilibrium for the group since the robots perceive a slight traction toward the center of the group itself, which makes them move in a circle around it. The experiments show that the stronger the symmetry of the group with respect to its center, the more likely that it will fall into this stable state.

The illustrated robots' behavior indicates that the distributed coordination performed by the evolved robots' controller relies on the self-organizing mechanism of positive feedback. Indeed, the behavior that the robots exhibit at the individual level is of the type "conform to the behavior of the group," as requested by the positive

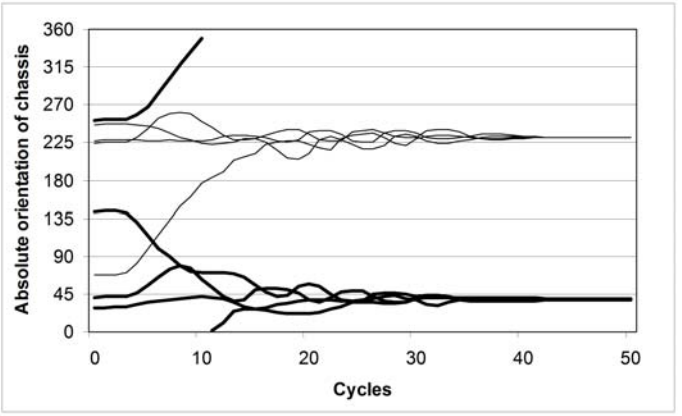


Fig. 7.5. The absolute angles (with respect to the environment) of the chassis orientations of the four robots forming a group (y-axis) in two tests (respectively bold and thin curves) in which the initial orientations are randomly selected.

feedback mechanism (see Sec. 7.2.1). Moreover at the collective level, as illustrated in Fig. 7.5, this behavior leads the robots to amplify some random fluctuations that eventually move the system away from the initial symmetric state. As a consequence the system achieves a complete asymmetric ordered state corresponding to a very good alignment and coordination of the robots.

7.5 The Emergence of Organization vs. Noise: A Phase Transition?

This section presents some preliminary results that suggest that the organization generated by the self-organizing mechanisms presented in the previous sections might have some features in common with the organization observed in phase transitions of physical systems. Note that to achieve stability of the data, the tests reported in this section were carried out with a group of robots formed by far more individuals than composed the group with which the controller was evolved, precisely 36 (Fig. 7.5). This was possible because, as shown in detail elsewhere (Baldassarre et al. 2006, 2007a), the evolved controller has very good scaling properties owing to the self-organizing mechanisms it relies upon.

First of all, let us see how the entropy index was applied to the robotic system. The possible orientation angle of each robot within the range $[0°, 360°]$ (considered as the state space of the elements of the system) was divided into eight "cells" of $45°$ each. The $0°$ angle was set to correspond to $22.5°$ clockwise with respect to the absolute angle of one particular robot chosen as the "pivot" (the angles of the other robots were then computed anticlockwise with respect to this origin angle). Note that while the origin angle on the basis of which the cells are computed is arbitrary, the selection done here ensured that when the group achieved high coordination, the the robots'

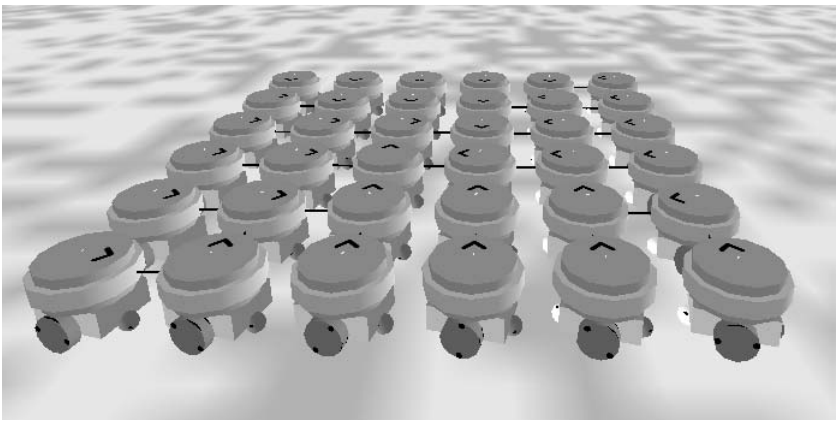


Fig. 7.6. A group of 36 robots engaged in the coordinated motion task. The black segments between the turrets of robot couples represent the physical connection between them.

chassis were located near to the center of the first cell and within it (minimum entropy). Moreover, as the pivot robot was always in the first cell, the number of microstates used to compute the entropy was computed with respect to $N - 1 = 35$ and not N robots.

In order to normalize E_m within $[0, 1]$, the scaling constant k of the index was set to one divided by the maximum value that $\ln[w_m]$ [see Eq. (7.1)] could assume for the system being studied, corresponding to a uniform distribution of the chassis orientations over the eight cells. In particular, given the small number of robots, for greater accuracy instead of considering Eq. (7.4) the maximum value was directly computed on the basis of Eq. 7.2. By considering the most uniform distribution that could be obtained with the 35 robots composing the system:

$$\begin{aligned} k &= 1/\ln[35!/(5!\ 5!\ 5!\ 4!\ 4!\ 4!\ 4!\ 4!)] \approx \\ &1/\ln[7.509 * 10^{26}] \approx 1/61.8843 \approx 0.01615 \end{aligned} \tag{7.6}$$

The graph in Fig. 7.7 illustrates the functioning of the index by reporting the level of entropy measured *during* 20 coordinated motion tests run with the system formed by the 36 robots shown in Fig. 7.6. The figure shows how the disorganization of the group initially decreases exponentially and then stabilizes at a null value when all the robots have converged to the same direction of motion. [See Baldassarre et al. (2007) for a statistical analysis and further consideration of these results.)

The tests designed to evaluate whether the self-organization of the robotic system has the properties of a phase transition relied upon a fine tuning of the ratio between noise and the signal returned by the traction sensor. (Recall from Sec. 7.3 that such a signal is used by the robots to "know" the direction of movement of the other robots so as to conform to it.) In particular, the noise/signal ratio was built through the following procedure (see Fig. 7.8): (a) At each time step, a 2D vector similar to the signal's vector was randomly generated; this vector had a random direction and a length ranging in $[0, 1]$. (b) The controller of the robot was fed with a vector equal to a weighted average

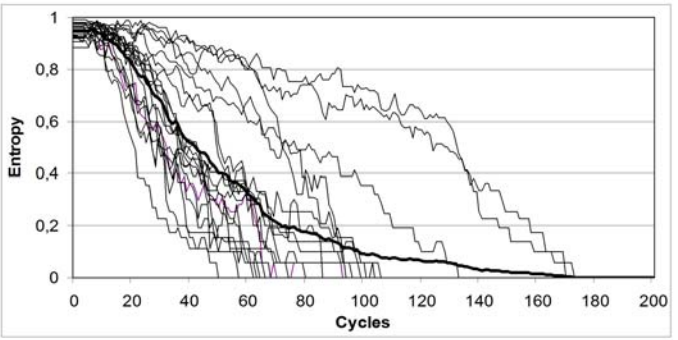


Fig. 7.7. Entropy of a group formed by 36 robots engaged in a coordinated motion task. The thin lines refers to the entropy measured in 20 tests each lasting for 200 cycles and were run with different initial random orientations of the robots' chassis; the bold line is the average of the 20 tests.

of the random vector and the signal vector; this average vector was obtained by multiplying the length of the two vectors by the respective "weights" of the average, and then computing the sum of the resulting vectors with the parallelogram rule. (c) The weights of this weighted average were equal to $\varepsilon \in [0, 1]$ and to $(1 - \varepsilon)$ for the noise and the signal, respectively: the "noise/signal ratio" manipulated in the experiments presented below was ε.

This computation of the ratio allowed running 20 tests with the 36-robot system where the noise/signal ratio ε was linearly lowered from one to zero during $20,000$ time steps. During these tests the entropy of the group was measured. Figure 7.9 reports the results of these measurements in terms of the relationship between the noise/signal ratio and the level of *order* of the group (i.e., the complement to one of the normalized entropy index). A first relevant fact highlighted by the figure is that the system starts to organize at a very high level of noise/signal ratio, about 0.8, indicating a surprising

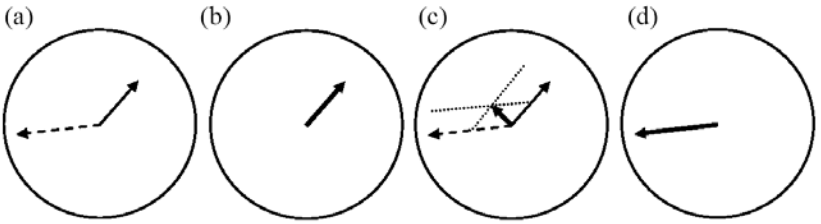


Fig. 7.8. Scheme of how the signal perceived by each robot was corrupted by noise at each time step of the tests depending on the noise/signal ratio: (a) an example of traction signal (continuous arrow) and noise (dashed arrow) represented as vectors; (b) if the ratio is equal to zero, the signal is not corrupted by noise (the signal perceived by the robot is represented by the bold arrow); (c) if the ratio has an intermediate value, for example 0.5 as in this case, the signal is partially corrupted by noise; (d) if the ratio is equal to one, the signal is completely substituted by noise.

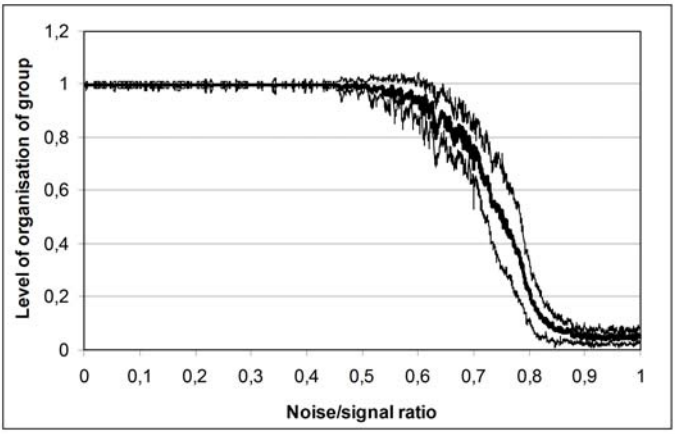


Fig. 7.9. Relationship between the noise/signal ratio and the level of organization of the group (equal to the complement to one of the normalized entropy) measured while slowly lowering the noise/signal ratio from one to zero. Average (bold line) $\pm$ standard deviation (thin lines) of the results obtained in 20 replications of the experiment.

robustness vs. noise of the self-organizing mechanisms employed by the system. Previous work (Baldassarre et al. 2006) had already indicated such a direction, but this result overcomes prior expectations and furnishes a quantitative measure of the level of such robustness.

The second relevant fact is that when the noise/signal ratio is progressively lowered, organization does not increase linearly but rather reaches its maximum level quite abruptly in correspondence with levels of noise/signal ratio ranging approximately between 0.6 and 0.8. This suggests that there is a critical noise/signal level in correspondence with which the system exhibits a transition from a disorganized to an organized state.

To further investigate the possible existence of such a critical value, groups of 20 tests were carried out by setting the noise/signal level to fixed values chosen in the range between 0.9 and 0.6, at intervals of 0.05, and by measuring the level of entropy of the system in $10,000$ cycles of simulation. The goal of these tests was to determine if there was a critical level of noise/signal ratio above and below which the system exhibited a discontinuous behavior in terms of overall organization. The outcome suggested that this might be the case. In particular, Fig. 7.10, which shows the outcome of these tests for three levels of noise/signal ratio, indicates that this critical level might be within $(0.75, 0.80)$. In fact, if the noise/signal value is set at 0.80, the entropy of the system fluctuates in the range of $(0.80, 1.00)$, i.e., around its maximum values. (In evaluating the level of order corresponding to such noise/signal values, consider that a level of entropy of 0.9 corresponds to quite uniform distributions of the robots on the cells, e.g.: $5, 6, 6, 6, 6, 5, 1, 0$.) On the other hand, for noise/signal values set at 0.75 in 18 out of 20 experiments the entropy level of the system initially decreases from about 0.95 to about 0.55, indicating that the system self-organizes and then stabilizes

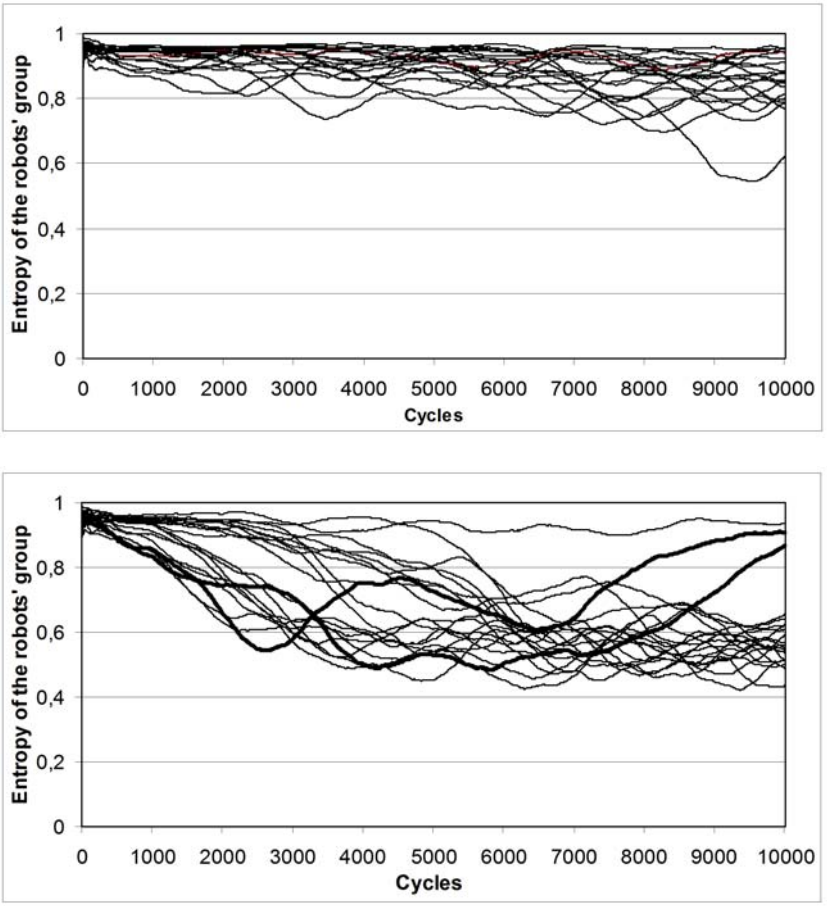


Fig. 7.10. Level of entropy (100-step moving average) of the 36-robot system in 20 tests lasting $10,000$ steps each, when the noise/signal ratio is set at two different fixed levels, namely 0.80 and 0.75 for the top and bottom graph, respectively. (The level of the noise/signal ratio is indicated on the y-axis of each graph by the bold arrow.) The two bold lines of the bottom graph refer to two tests where the system first reached an ordered state and then lost it.

at values ranging in $(0.45, 0.65)$. (In evaluating the level of order corresponding to such noise/signal values, consider that a level of entropy of 0.55 corresponds to quite concentrated distributions of the robots on the cells, e.g.: $0, 1, 6, 20, 7, 1, 0, 0$.) Once the system "gets locked" into the ordered state, it tends to resist noise perturbations, as predicted by the considerations presented in Section 7.2.1. Indeed, entropy rose again to high values in only 2 out of 20 cases after the system reached the ordered state (see the bold lines in the bottom graph of Fig. 7.10).

7.6 Conclusions

This chapter presented an evolved multirobot system with decentralized control that is capable of achieving coordination in order to accomplish a collective task on the basis of self-organizing mechanisms. These mechanisms were first described at the levels of individual and collective behavior, and then the effects they produced on the level of organization of the system were quantitatively analyzed on the basis of an index based on Boltzmann entropy. This analysis showed that when one slowly decreases the noise/signal ratio related to the signal that the robots use to coordinate, the dynamics of the self-organization exhibited by the system resembles the self-organization characterizing physical systems undergoing phase transitions. In particular, the order of the system tends to emerge quite abruptly when the ratio is lowered below a critical value.

If confirmed, the hypothesis that the dynamics of the level of order of self-organized multirobot systems might have the features of a phase transition would have important implications. In fact, it would imply that self-organization of collective systems tends to manifest in an all-or-nothing fashion depending on the quality of the signals exchanged by the elements forming the system. Moreover, when such quality overcomes a critical value, by even a *small* amount, the organization produced by the self-organizing mechanisms becomes fully effective and robust vs. noise (as the system "locks in" in its state of order). These implications are relevant for engineering purposes. For example, identifying the critical noise-signal level that characterizes a distributed multirobot system might allow for adjusting of the physical setup of the latter so as to achieve a reliable level of robustness of its self-organization. The implications are also important for scientific purposes, e.g., for investigating self-organization in collective biological systems (Bonabeau et al. 1999; Camazine et al. 2001; Anderson et al. 2002). In fact in some systems of this kind self-organization emerges quite suddenly if some parameters of the system change beyond certain thresholds. For example, trail formation in ants requires that the number of ants that compose the group, and hence the amount of pheromone released on the ground, reach a certain level for the organization of the group to emerge. In fact, given that the laid pheromone trace slowly vanishes with time, if the number of ants, and hence the level of the released pheromone, is not high enough, the signal that it furnishes to the ants is too weak to allow them to self-organize.

The added value of the chapter resides in the techniques it presented. In particular, such techniques might not only be used to measure the level of organization of decentralized (and centralized) systems, as was done here, but might also be used directly as a fitness function to evolve systems that exhibit useful behaviors (for some examples that use entropy indexes differently from the way they are used here, see Prokopenko et al. 2006) or to explore the self-organization potential of systems. Moreover, the identification of the critical noise/signal ratio that characterizes a decentralized robotic system might be a way to furnish a quantitative *measure of the robustness* of the self-organizing principles that govern it.

Notwithstanding the relevance of all these implications, it is important to note that the results presented here, in particular those related to the hypothesis according to which under some conditions self-organization of some multirobot systems might behave as a phase transition, are in many respects preliminary. For example, further

research is needed to corroborate or reject the hypothesis itself, to better understand the behavior of the system in correspondence with the critical level of the noise/signal ratio, and to better understand the relationship that exists between the level of order of the system and the role that it plays in its functioning (e.g., in its capacity to displace in space). Moreover, it might be useful to build a mathematical abstract model of the system to carry out an analytical study to ascertain at a more formal level if it possesses the properties that characterize phase transitions. For example, this analysis might identify some quantities associated with the self-organization of the robotic system that behave similarly to "free energy" or "latent heat" in phase transitions of physical systems. (For an introduction to these topics, see http://en.wikipedia.org/wiki/Phase_transition.)

A final observation is that experiments similar to those conducted here by slowly lowering the noise/signal ratio might be also conducted on the actuator's noise and on the controller's effectiveness. In this regard it might be possible to envisage a way to regulate the "noise/effectiveness level" of actuators, or the "level of effectiveness" of the controller in ways similar to the one used here to regulate the noise/signal ratio of sensors. These experiments might show that these two manipulations also lead to phase transitions at the level of the system's overall organization.

Acknowledgments

This research has been supported by the *SWARM-BOTS* project funded by the Future and Emerging Technologies program (IST-FET) of the European Commission under grant IST-2000-31010. I thank Stefano Nolfi and Domenico Parisi with which I designed, developed and studied extensively the robotic setup used in the paper.

References

Anderson, P. (1997). *Basic Notions of Condensed Matter Physics*. Perseus, Cambridge, MA.

Anderson, C., Theraulaz, G., and Deneubourg, J.-L. (2002). Self-assemblages in insect societies. *Insectes Sociaux,* 49:1–12.

Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., and Nolfi, S. (2007a). Self-organised coordinated motion in groups of physically connected robots. *IEEE Transactions in Systems, Man and Cybernetics* 37(1):224–239.

Beckers, R., Holland, O. E., and Deneubourg, J.-L. (1994). From local actions to global tasks: Stigmergy and collective robotics. In Brooks, R. A., and Maes, P., editors, *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV),* pages 181–189. MIT Press, Cambridge, MA.

Baldassarre, G., Nolfi, S., and Parisi D. (2003). Evolution of collective behaviour in a group of physically linked robots. In Raidl, G., Guillot, A., and Meyer, J.-A., editors, *Applications of Evolutionary Computing - Proceedings of the Second European Workshop on Evolutionary Robotics,* pages 581-592. Springer, Berlin.

Baldassarre, G., Parisi, D., and Nolfi, S. (2006). Distributed coordination of simulated robots based on self-organization. *Artificial Life,* 12(3):289–311.

Baldassarre, G., Parisi, D., and Nolfi, S. (2007a). Measuring coordination as entropy decrease in groups of linked simulated robots. In Minai, A., and Bar-Yam, Y. editors, *Proceedings of the Fifth International Conference on Complex Systems (ICCS2004)*. Springer, Berlin, in press.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: From Natural to Artificial Systems*. Oxford University Press, New York.

Camazine, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2001). *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ.

Cao, Y. U., Fukunaga, A. S., and Kahng, A. B. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:1–23.

Dorigo, M., and Sahin, E. (2004). Swarm robotics: Special issue editorial. *Autonomous Robots,* 17(2-3):111–113.

Dorigo, M., Trianni, V., Sahin, E., Gross, R., Labella, T. H., Baldassarre, G., Nolfi, S., Denebourg, J-L, Floreano, D., and Gambardella, L. M. (2004). Evolving self-organizing behavior for a swarm-bot. *Autonomous Robots,* 17(2-3):223–245.

Dudek, G., Jenkin, M., Milios E., and Wilkes, D. (1996). A taxonomy for multi-agent robotics. *Autonomous Robots,* 3:375–397.

Feldman, P. D. (1998). A brief introduction to information theory, excess entropy and computational mechanics. Technical report. Department of Physics, University of California.

Holland, O., and Melhuish, C. (1999). Stimergy, self-organization, and sorting in collective robotics. *Artificial Life,* 5:173–202.

Ijspeert, A. J., Martinoli, A., Billard, A., Gambardella, L. M. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots,* 11:149–171.

Krieger, M. J. B., Billeter, J. B., and Keller, L. (2000). Ant-like task allocation and recruitment in cooperative robots. *Nature,* 406:992–995.

Kube, R. C., and Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and autonomous systems*, 30:85–101.

Kube, C. R., and Zhang, H. (1993). Collective robotics: From social insects to robots, *Adaptive Behavior,* 2(2):189–219.

Mondada, F., Pettinaro, G., Guignard, A., Kwee, I., Floreano, D., Denebourg, J-L, Nolfi, S., Gambardella, L. M., and Dorigo, M. (2004). Swarm-bot: A new distributed robotic concept. *Autonomous Robots*, 17(2-3):193–221.

Nolfi, S., and Floreano, D. (2001). *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA.

Prokopenko, M., Boschetti, F., and Ryan, A. J. (2007). An information-theoretic primer on complexity, self-organisation and emergence. *Advances in Complex Systems*, submitted.

Prokopenko, M., Gerasimov, V., and Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J., Marocco, D., Meyer, J.-A., Miglino, O., Parisi, D., editors, *From Animals to Animats 9: Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior (SAB-2006)*, volume 4095 of *Lecture Notes in Computer Science*, pages 558–569. Springer, Berlin.

Quinn, M., Smith, L., Mayley, G., and Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, 361:2321–2344.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model, *Computer Graphics*, 21(4):25–34.

Trianni, V., Nolfi, S., and Dorigo, M. (2006). Cooperative hole-avoidance in a swarm-bot. *Robotics and Autonomous Systems*, 54(2):97–103.

Tsai, S., and Salinas, S.R. (1998). Fourth-order cumulants to characterize the phase transitions of a spin-1 Ising model. *Brazilian Journal of Physics*, 28(1):58–65.

8

Distributed Control of Microscopic Robots in Biomedical Applications

Tad Hogg

8.1 Microscopic Robots

The development of molecular electronics, motors, and chemical sensors could enable the construction of large numbers of devices able to sense, compute, and act in micron-scale environments. Such microscopic robots, of sizes comparable to bacteria, could simultaneously monitor entire populations of cells individually in vivo. Their small size allows the robots to move through the tiniest blood vessels, so they would be able to pass within a few cell diameters of most of the cells in large organisms via their circulatory systems to perform a wide variety of biological research and medical tasks. For instance, robots and nanoscale-structured materials inside the body could significantly improve disease diagnosis and treatment (Freitas 1999; Keszler et al. 2001; Morris 2001; NIH 2003). Initial tasks for microscopic robots include in vitro research via simultaneous monitoring of chemical signals exchanged among many bacteria in a biofilm. The devices could also operate in multicellular organisms as passively circulating sensors. Such devices, with no need for locomotion, would detect programmed patterns of chemicals as they pass near cells. More advanced technology could create devices able to communicate to external detectors, allowing real-time in vivo monitoring of many cells. The devices could also have the capability of acting on their environment, e.g., releasing drugs at locations with specific chemical patterns or mechanically manipulating objects for microsurgery. Extensive development and testing is necessary before clinical use, first for high-resolution diagnostics and later for programmed actions at cellular scales.

Realizing these benefits requires fabricating the robots cheaply, in large numbers and with sufficient capabilities. Such fabrication is beyond current technology. Nevertheless, ongoing progress in engineering nanoscale devices could eventually enable production of such robots. One approach to creating microscopic programmable machines is engineering biological systems, e.g., bacteria executing simple programs (Andrianantoandro et al. 2006) and DNA computers responding to logical combinations of chemicals (Benenson et al. 2004). However, biological organisms have limited material properties and computational speed. Instead we focus on machines based on plausible extensions of current molecular-scale electronics, sensors, and

motors (Howard 1997; Collier et al. 1999; Montemagno and Bachand 1999; Craighead 2000; Fritz et al. 2000; Soong et al. 2000; Berna et al. 2005; Wang and Williams 2005). These devices could provide components for stronger and faster microscopic robots than is possible with biological organisms. Thus the focus here is on nonbiological robots containing nanoscale sensors and electronics, along with a power source, within a protective shell. As technology improves, such robots could be supplemented with other capabilities such as communication and locomotion.

As we cannot yet fabricate microscopic robots with molecular electronics components, estimates of their performance rely on plausible extrapolations from current technology. The focus in this chapter is on robots for biomedical applications requiring only modest hardware capabilities, which will be easier to fabricate than more capable robots. Designing controls for microscopic robots is a key challenge: not only enabling useful performance but also compensating for their limited computation, locomotion, or communication abilities. Distributed control is well suited to these capabilities as it emphasizes locally available information and achieves overall objectives through self-organization of the collection of robots. Theoretical studies allow developing such controls and estimating their performance prior to fabrication, thereby indicating design trade-offs among hardware capabilities, control methods, and task performance. Such studies of microscopic robots complement analyses of individual nanoscale devices (McCurdy et al. 2002; Wang and Williams 2005) and indicate that even modest capabilities enable a range of novel applications.

The operation of microscopic robots differs significantly from that of larger robots (Mataric 1992), especially for biomedical applications. First, the physical environment is dominated by viscous fluid flow. Second, thermal noise is a significant source of sensor error and Brownian motion limits the ability to follow precisely specified paths. Third, relevant objects are often recognizable via chemical signatures rather than, say, visual markings or specific shapes. Fourth, the tasks involve large numbers of robots, each with limited abilities. Moreover, a task will generally only require a modest fraction of the robots to respond appropriately, not for all, or even most, robots to do so. Thus controls using random variations are likely to be effective simply due to the large number of robots. This observation contrasts with teams of larger robots with relatively few members: incorrect behavior by even a single robot can significantly decrease team performance. These features suggest that reactive distributed control is particularly well-suited for microscopic robots.

Organisms contain many microenvironments, with distinct physical, chemical, and biological properties. Often, precise quantitative values of properties relevant for robot control will not be known a priori. This observation suggests a multistage protocol for using the robots. First, an information-gathering stage with passive robots placed into the organism, e.g., through the circulatory system, to measure relevant properties (Hogg and Kuekes 2006). The information from these robots, in conjunction with conventional diagnostics at larger scales, could then determine appropriate controls for further actions in subsequent stages of operation.

For information gathering, each robot notes in its memory whenever chemicals matching a prespecified pattern are found. Eventually, the devices are retrieved and the information in their memories is extracted for further analysis in a conventional

computer with far more computational resources than are available to any individual microscopic robot. This computer would have access to information from many robots, allowing evaluation of aggregate properties of the population of cells that individual robots would not have access to, e.g., the number of cells presenting a specific combination of chemicals. This information allows estimating the spatial structure and strength of the chemical sources. The robots could detect localized high concentrations that are too low to distinguish from background concentrations when diluted in the whole blood volume as obtained with a sample. Moreover, if the detection consists of the joint expression of several chemicals, each of which also occurs from separate sources, the robot's pattern recognition capability could identify the spatial locality, which would not be apparent when the chemicals are mixed throughout the blood volume.

Estimating the structure of the chemical sources from the microscopic sensor data is analogous to computerized tomography (Natterer 2001). In tomography, the data consist of integrals of the quantity of interest (e.g., density) over a large set of lines with known geometry selected by the experimenter. The microscopic sensors, on the other hand, record data points throughout the tissue, providing more information than just one aggregate value such as the total number of events. However, the precise path of each sensor through the tissue, i.e., which vessel branches it took and the locations of those vessels, will not be known. This mode of operation also contrasts with uses of larger distributed sensor networks able to process information and communicate results while in use.

Actions based on the information from the robots would form a second stage of activity, perhaps with specialized microscopic robots (e.g., containing drugs to deliver near cells), with controls set based on the calibration information retrieved earlier. For example, the robots could release drugs at chemically distinctive sites (Freitas 1999, 2006) with specific detection thresholds determined with the information retrieved from the first stage of operation. Or robots could aggregate at the chemical sources (Casal et al. 2003; Hogg 2006) or manipulate biological structures based on surface chemical patterns on cells, e.g., as an aid for microsurgery in repairing injured nerves (Hogg and Sretavan 2005). These active scenarios require more advanced robot capabilities, such as locomotion and communication, than are needed for passive sensing. The robots could monitor environmental changes due to their actions, thereby documenting the progress of the treatment. Thus the researcher or physician could monitor the robots' progress and decide whether and when they should continue on to the next step of the procedure. Using a series of steps, with robots continuing with the next step only when so instructed by the supervising person, maintains overall control of the robots and simplifies the control computations that each robot must perform itself.

To illustrate controls for large collections of microscopic robots, this chapter considers a prototypical diagnostic task of finding a small chemical source in a multicellular organism via the circulatory system. To do so, we first review plausible capabilities for microscopic robots and the physical constraints due to operation in fluids at a low Reynolds number, diffusion-limited sensing, and thermal noise from Brownian motion. We then discuss techniques for evaluating the behavior of large collections of robots and examine a specific task scenario. The emphasis here is on feasible

performance with plausible biophysical parameters and robot capabilities. Evaluation metrics include minimizing hardware capabilities to simplify fabrication and ensuring safety, speed, and accuracy for biological research or treatment in a clinical setting.

8.2 Capabilities of Microscopic Robots

This section describes plausible robot capabilities based on currently demonstrated nanoscale technology. Minimal capabilities needed for biomedical tasks include chemical sensing, computation, and power. Additional capabilities, enabling more sophisticated applications, include communication and locomotion.

8.2.1 Chemical Sensing

Large-scale robots often use sonar or cameras to sense their environment. These sensors locate objects from a distance and involve sophisticated algorithms with extensive computational requirements. In contrast, microscopic robots for biological applications use mainly chemical sensors, e.g., the selective binding of molecules to receptors altering the electrical characteristics of nanoscale wires. The robots could also examine chemicals inside nearby cells (Xie et al. 2006).

Microscopic robots and bacteria face similar physical constraints in detecting chemicals (Berg and Purcell 1977). The diffusive capture rate γ for a sphere of radius a in a region with concentration C is (Berg 1993)

$$\gamma = 4\pi DaC, \tag{8.1}$$

where D is the diffusion coefficient of the chemical. Even when sensors cover only a relatively small fraction of the device surface, the capture rate is almost this large (Berg 1993). Nonspherical devices have similar capture rates so Eq. (8.1) is a reasonable approximation for a variety of designs.

Current molecular electronics (Wang and Williams 2005) and nanoscale sensors (Li et al. 2005; Patolsky and Lieber 2005; Sheehan and Whitman 2005) indicate plausible sensor capabilities. At low concentrations, sensor performance is primarily limited by the time it takes for molecules to diffuse to the sensor, and statistical fluctuations in the number of molecules encountered is a major source of noise. Moreover, other chemicals with similar binding properties to the chemical of interest would give rise to additional noise. Thus the selectivity of the sensor is important in setting the noise level, and may trade-off with the time used to determine whether a detection occurred (Alon 2007).

8.2.2 Timing and Computation

With the relevant fluid speeds and chemical concentrations described in Section 8.4, robots pass through high concentrations near individual cells on millisecond time scales. Thus identifying significant clusters of detections due to high concentrations

requires a clock with millisecond resolution. This clock need not be globally synchronized with other devices.

In a simple scenario, devices just store sensor detections in their memories for later retrieval. In this case most of the computation to interpret sensor observations takes place in larger computers after the devices are retrieved. Recognizing and storing a chemical detection involves at least a few arithmetic operations to compare sensor counts to threshold values stored in memory. An estimate on the required computational capability is about 100 elementary logic operations and memory accesses within a 10-ms measurement time, which gives about 10^4 logic operations per second. While modest compared to current computers, this rate is significantly faster than demonstrated for programmable bacteria (Andrianantoandro et al. 2006) but well within the capabilities of molecular electronics.

8.2.3 Communication

A simple form of one-way communication is robots passively sensing electromagnetic or acoustic signals from outside the body. An example is using radio frequency to produce local heating in metal nanospheres attached to the devices (Hamad-Schifferli et al. 2002). Such signals could activate robots only within certain areas of the body at, say, centimeter-length scales. Using external signals to localize the robots more precisely is difficult due to variations in propagation through tissues, so such localization requires more complex technology (Freitas 1999) than we consider in this discussion.

Additional forms of communication—between nearby robots and sending information to detectors outside the organism—are more difficult to fabricate and require significant additional power. For example, since the robots considered here have chemical sensors, a natural approach to communication would be to use onboard storage of specific chemicals they could release for detection by other nearby robots. Such diffusion-mediated signals are not effective for communicating over distances beyond a few microns, but could mark the environment for detection by other robots that pass nearby later, i.e., stigmergy (Bonabeau et al. 1999). Acoustic signals provide more versatile communication. Compared to fluid flow, acoustic signals are essentially instantaneous, but power constraints limit their range to about $100\,\mu$m (Freitas 1999).

8.2.4 Locomotion

Biomedical applications will often involve robots operating in fluids. Viscosity dominates the robot motion, with different physical behaviors than for larger organisms and robots (Purcell 1977; Vogel 1994; Fung 1997; Karniadakis and Beskok 2002; Squires and Quake 2005). The ratio of inertial to viscous forces for an object of size s moving with velocity v through a fluid with viscosity η and density ρ is the Reynolds number $\mathrm{Re} \equiv s\rho v/\eta$. Using typical values for density and viscosity (e.g., of water or blood plasma) in Table 8.1 and noting that reasonable speeds for robots with respect to the fluid (Freitas 1999) are comparable to the fluid flow speed in small vessels (i.e., ~ 1 mm/s), we find that the motion of a 1-μ robot has $\mathrm{Re} \approx 10^{-3}$, so viscous forces dominate. Consequently, robots applying a locomotive force quickly reach terminal

Table 8.1. Parameters for the environment, robots, and the chemical signal.

Parameter	value
Tissue, Vessels and Source	
Vessel radius	$R = 5\ \mu$m
Vessel length	$L = 1000\ \mu$m
Number density of vessels in tissue	$\rho_{\rm vessel} = 5 \times 10^{11}/{\rm m}^3$
Tissue volume	$V = 10^{-6}\ {\rm m}^3$
Source length	$L_{\rm source} = 30\ \mu$m
Fluid	
Fluid density	$\rho = 10^3\ {\rm kg/m}^3$
Fluid viscosity	$\eta = 10^{-3}\ {\rm kg/m/s}$
Average fluid velocity	$v_{\rm avg} = 10^{-3}{\rm m/s}$
Fluid temperature	$T = 310$ K
Robots	
Robot radius	$a = 1\ \mu$m
Number density of robots	$\rho_{\rm robot} = 2 \times 10^{11}\ {\rm robot/m}^3$
Robot diffusion coefficient	$D_{\rm robot} = 7.6 \times 10^{-14}\ {\rm m}^2/{\rm s}$
Chemical signal	
Production flux at target	$F_{\rm source} = 5.6 \times 10^{13}\ {\rm molecule}s/{\rm s/m}^2$
Diffusion coefficient	$D = 10^{-10}\ {\rm m}^2/{\rm s}$
Concentration near source	$C_{\rm source} = 1.8 \times 10^{18}\ {\rm molecule}s/{\rm m}^3$
Background concentration	$C_{\rm background} = 6 \times 10^{15}\ {\rm molecule}s/{\rm m}^3$

The robots are spheres with radius a. The chemical signal concentrations correspond to a typical 10-kilodalton chemokine molecule, with mass concentrations near the source and background (i.e., far from the source) equal to $3 \times 10^{-5}\ {\rm kg/m}^3$ and $10^{-7}\ {\rm kg/m}^3$, respectively. $C_{\rm source}$ is equivalent to a 3 nM solution. The $\rho_{\rm vessel} V$ small vessels in the tissue volume occupy a fraction $\rho_{\rm vessel}\pi R^2 L \approx 4\%$ of the volume.

velocity in the fluid, i.e., applied force is proportional to velocity as opposed to the more familiar proportionality to acceleration of Newton's law $F = ma$. By contrast, a swimming person has an Re about a billion times larger.

Flow in a pipe of uniform radius R has a parabolic velocity profile: velocity at a distance r from the axis is

$$v(r) = 2v_{\rm avg}(1 - (r/R)^2), \tag{8.2}$$

where $v_{\rm avg}$ is the average speed of fluid in the pipe.

Robots moving through the fluid encounter significant drag. For instance, an isolated sphere of radius a moving at speed v through a fluid with viscosity η has a drag force

$$6\pi a\eta v. \tag{8.3}$$

Although not quantitatively accurate near boundaries or other objects, this expression estimates the drag in those cases as well. For instance, a numerical evaluation of drag force on a 1-μm-radius sphere moving at velocity v with respect to the fluid flow near the center of a 5-μm-radius pipe has drag about three times larger than is given by Eq. (8.3). Other reasonable choices for robot shape have similar drag.

Fluid drag moves robots in the fluid. An approximation is that robots without active locomotion move with the same velocity as fluid would have at the center of the robot if the robot were not there. Numerical evaluation of the fluid forces on the robots for the parameters of Table 8.1 show that the robots indeed move close to this speed when the spacing between them is many times their size. Closer packing leads to more complex motion owing to hydrodynamic interactions (Hernandez-Ortiz et al. 2005; Riedel et al. 2005).

8.2.5 Additional Sensing Capabilities

In addition to chemical sensing, robots could sense other properties to provide high-resolution spatial correlation of various aspects of their environment. For example, nanoscale sensors for fluid motion can measure fluid flow rates at speeds relevant for biomedical tasks (Ghosh et al. 2003), allowing robots to examine in vivo microfluidic behavior in small vessels. In particular, at low Re, boundary effects extend far into the vessel (Squires and Quake 2005), giving an extended gradient in fluid speed with higher fluid shear rates nearer the wall. Thus, several such sensors, extending a short distance from the device surface in various directions, could estimate shear rates and hence the direction to the wall or changes in the vessel geometry. Another example for additional sensing is optical scattering in cells, as has been demonstrated to distinguish some cancer from normal cells in vitro (Gourley et al. 2005).

8.2.6 Power

To estimate the power for robot operation, each logic operation in current electronic circuits uses 10^4–10^5 times the thermal noise level $k_BT = 4 \times 10^{-21}$J at the fluid temperature of Table 8.1, where k_B is the Boltzmann constant. Near-term molecular electronics could reduce this to $\approx 10^3 k_BT$, in which case 10^4 operations per second uses a bit less than 0.1 pW. Additional energy will be needed for signals within the computer and with its memory.

This power is substantially below that required for locomotion or communication. For instance, due to fluid drag and the inefficiencies of locomotion in viscous fluids, robots moving through the fluid at ≈ 1 mm/s dissipate 1 pW (Berg 1993). However, these actions may operate only occasionally, and for short periods, when the sensor detects a signal, whereas computation could be used continuously while monitoring for such signals.

For tasks of limited duration, an on-board fuel source created during manufacture could suffice. Otherwise, the robots could use energy available in their environment, such as converting vibrations to electrical energy (Wang and Song 2006)

or chemical generators. Typical concentrations of glucose and oxygen in the bloodstream could generate ≈ 1000 pW continuously, limited primarily by the diffusion rate of these molecules to the device (Freitas 1999). For comparison, a typical person at rest uses about 100 Ws. Beyond simply generating the required power, the robots require effective machines to distribute and use the energy, with several possible approaches (Freitas 1999).

8.3 Evaluating Collective Robot Performance

As the microscopic robots cannot yet be fabricated and quantitative biophysical properties of many microenvironments are not precisely known, performance studies must rely on plausible models of both the machines and their task environments (Drexler 1992; Freitas 1999; Requicha 2003). Microorganisms, which face physical microenvironments similar to those of future microscopic robots, provide some guidelines for feasible behaviors.

Cellular automata constitute one technique for evaluating collective robot behavior. For example, a two-dimensional scenario shows how robots might assemble structures (Arbuckle and Requicha 2004) using local rules. Such models can help understand structures formed at various scales through simple local rules and some random motions (Whitesides and Grzybowski 2002; Griffith et al. 2005). However, cellular automata models either ignore or greatly simplify physical behaviors such as fluid flow. Another analysis technique considers swarms (Bonabeau et al. 1999), which are well-suited to microscopic robots with their limited physical and computational capabilities and large numbers. Most swarm studies focus on macroscopic robots or behaviors in abstract spaces (Gazi and Passino 2004) that do not specifically include physical properties unique to microscopic robots. In spite of the simplified physics, these studies show how local interactions among robots lead to various collective behaviors and provide broad design guidelines.

Simulations including physical properties of microscopic robots and their environments can evaluate performance of robots with various capabilities. Simple models, such as a two-dimensional simulation of chemotaxis (Dhariwal et al. 2004), provide insight into how robots find microscopic chemical sources. A more elaborate simulator (Cavalcanti and Freitas 2002) includes three-dimensional motions in viscous fluids, Brownian motion, and environments with numerous cell-sized objects, though without accounting for how they change the fluid flow. Studies of hydrodynamic interactions (Hernandez-Ortiz et al. 2005) among moving devices include more accurate fluid effects.

Another approach to robot behaviors employs a stochastic mathematical framework for distributed computational systems (Hogg and Huberman 2004; Lerman et al. 2001). This method directly evaluates average behaviors of many robots through differential equations determined from the state transitions used in the robot control programs. Direct evaluation of average behavior avoids the numerous repeated runs of a simulation needed to obtain the same result. This approach is best suited for simple control strategies, with minimal dependencies on events in individual robot histories.

Microscopic robots, with limited computational capabilities, will likely use relatively simple reactive controls, for which this analytic approach is ideally suited. Moreover, these robots will often act in environments with spatially varying fields, such as chemical concentrations and fluid velocities. Even at micron scales, the molecular nature of these quantities can be approximated as continuous fields with behavior governed by partial differential equations. For application to microscopic robots, this approximation extends to the robots themselves, treating their locations as a continuous concentration field, and their various control states as corresponding to different fields, much as multiple reacting chemicals are described by separate concentration fields. This continuum approximation for average behavior of the robots will not be as accurate as when applied to chemicals or fluids, but nevertheless gives a simple approach to average behaviors for large numbers of robots responding to spatial fields. One example of this approach is following chemical gradients in one dimension without fluid flow (Galstyan et al. 2005).

Cellular automata, swarms, physically based simulations, and stochastic analysis are all useful tools for evaluating the behaviors of microscopic robots. One example is evaluating the feasibility of rapid, fine-scale response to chemical events too small for detection with conventional methods, including sensor noise inherent in the discrete molecular nature of low concentrations. This chapter examines this issue in a prototypical task using the stochastic analysis approach. This method allows incorporating more realistic physics than is used with cellular automata studies and is computationally simpler than repeated simulations to obtain average behaviors. The technique is limited in requiring approximations for dependencies introduced by the robot history, but readily incorporates physically realistic models of sensor noise and consequent mistakes in the robot control decisions. The stochastic analysis indicates plausible performance, on average, and thereby suggests scenarios suited for further, more detailed, simulation studies.

8.4 A Task Scenario

As a prototypical task for microscopic robots, we consider their ability to respond to a cell-sized source releasing chemicals into a small blood vessel. This scenario illustrates a basic capability for the robots: identifying small chemically distinctive regions, with high sensitivity, owing to their ability to get close (within a few cell diameters) to a source. This capability would be useful as part of more complex tasks where the robots are to take some action at such identified regions. Even without additional actions, the identification itself would provide extremely accurate and rapid diagnostic capability compared to current technology.

Microscopic robots acting independently to detect specific patterns of chemicals are analogous to swarms (Bonabeau et al. 1999) used in foraging, surveillance, or search tasks. Even without locomotion capabilities, large numbers of such devices could move through tissues by flowing passively in moving fluids, e.g., through blood vessels or with lymph fluid. As the robots move, they can monitor for preprogrammed patterns of chemical concentrations.

Chemicals with high concentrations are readily detected with the simple procedure of analyzing a blood sample. Thus the chemicals of interest for microscopic robot applications will generally have low concentrations. With sufficiently low concentrations and small sources, the devices are likely to only encounter a few molecules while passing near the source, leading to significant statistical fluctuations in the number of detections.

We can consider this task from the perspective of stages of operation discussed in the introduction. For a diagnostic first stage, the robots need only store events in their memory for later retrieval, at which point a much more capable conventional computer can process the information. For an active second stage in which robots react to their detections, e.g., to aggregate at a source location or release a drug, the robots would need to determine themselves when they are near a suitable location. In this subsequent stage, the robots would need a simple control decision procedure, within the limits of local information available to them and their computational capacity. Such a control program could involve comparing counts to various thresholds, which were determined by analysis of a previous diagnostic stage of operation.

8.4.1 Example Task Environment

Tissue microenvironments vary considerably in their physical and chemical properties. As a specific example illustrating the capabilities of passive motion by microscopic sensors, we consider a task environment consisting of a macroscopic tissue volume V containing a single microscopic source producing a particular chemical (or combination of chemicals) while the rest of the tissue has this chemical at much lower concentrations. This tissue volume contains a large number of blood vessels, and we focus on chemical detection in the small vessels, since they allow exchange of chemicals with surrounding tissue and account for most of the surface area. A rough model of the small vessels has each with a length L and has them occurring throughout the tissue volume with number density ρ_{vessel}. Localization to volume V could be due to a distinctive chemical environment (e.g., high oxygen concentrations in the lungs), an externally supplied signal (e.g., ultrasound) detectable by sensors passing through vessels within the volume, or a combination of both. The devices are active only when they detect that they are in the specified region.

Robots moving with fluid in the vessels will, for the most part, be in vessels containing only the background concentration, providing numerous opportunities for incorrectly interpreting background concentration as source signals. These false positives are spurious detections due to statistical fluctuations from the background concentration of the chemical. Although such detections can be rare for individual devices, when applied to tasks involving small sources in a large tissue volume, the number of opportunities for false positive responses can be orders of magnitude larger than the opportunities for true positive detections. Thus even a low false positive rate can lead to many more false positive detections than true positives. The task scenario examined in this chapter thus includes estimating both true and false positive rates, addressing the question of whether simple controls can achieve a good trade-off of both a high true positive rate and a low false positive rate.

For simplicity, we consider a vessel containing only flowing fluid, robots, and a diffusing chemical arising from a source area on the vessel wall. This scenario produces a static concentration of the chemical throughout the vessel, thereby simplifying the analysis. We examine the rate at which robots find the source and the false positive rate as functions of the detection threshold used in a simple control rule computed locally by each robot.

Figure 8.1 shows the task geometry: a segment of the vessel with a source region on the wall emitting a chemical into the fluid. Robots continually enter one end of the vessel with the fluid flow. We assume that the robots have neutral buoyancy and move passively in the fluid, with a speed given by Eq. (8.2) at their centers. This approximation neglects the change in fluid flow due to the robots, but is reasonable for estimating detection performance when the robots are at a low enough density to be spaced apart by many times their size, as is the case for the example presented here. The robot density in Table 8.1 corresponds to 10^9 robots in the entire 5-liter blood volume of a typical adult, an example of medical applications using a huge number of microscopic robots (Freitas 1999). These robots use only about 10^{-6} of the vessel volume, far less than the 20–40% occupied by blood cells. The total mass of all the robots is about 4 mg.

The scenario for microscopic robots examined here is detecting small areas of infection or injury. The chemicals arise from the initial immunological response at the injured area and enter nearby small blood vessels to recruit white blood cells (Janeway et al. 2001). We consider a typical protein produced in response to injury, with concentration near the injured tissue of about $30\,\mathrm{ng/ml}$ and background concentration in the bloodstream about 300 times smaller. These chemicals, called chemokines, are proteins with molecular weight around 10^4 Da (equal to about 2×10^{-23} kg). These values lead to the parameters for the chemical given in Table 8.1, with chemical concentrations well above the demonstrated sensitivity of nanoscale chemical sensors (Patolsky and Lieber 2005; Sheehan and Whitman 2005). This example incorporates features relevant for medical applications: a chemical indicating an area of interest, diffusion into flowing fluid, and a prior background level of the chemical limiting sensor discrimination.

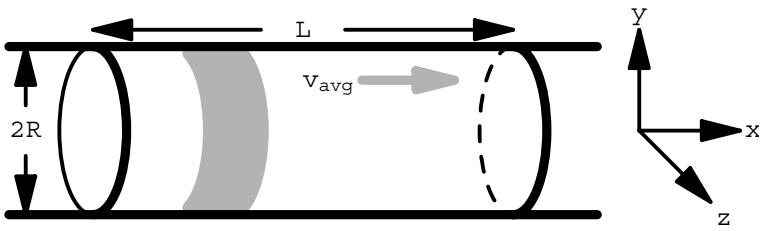


Fig. 8.1. Schematic illustration of the task geometry as a vessel of length L and radius R. Fluid flows in the positive x-direction with average velocity v_{avg}. The gray area is the source region wrapped around the surface of the pipe.

8.4.2 Diffusion of Robots and Chemicals

Diffusion arising from Brownian motion is somewhat noticeable for microscopic robots, and significant for molecules. The diffusion coefficient D, which depends on an object's size, characterizes the resulting random motion, with root-mean-square displacement of $\sqrt{6Dt}$ in a time t. For the parameters of Table 8.1, this displacement for the robots is about $0.7\sqrt{t}\,\mu$, with t measured in seconds. Brownian motion also randomly alters robot orientation.

The chemical concentration C is governed by the diffusion equation (Berg 1993)

$$\frac{\partial C}{\partial t} = -\nabla \cdot \mathbf{F}, \tag{8.4}$$

where $\mathbf{F} = -D\nabla C + \mathbf{v}C$ is the chemical flux, i.e., the rate at which molecules pass through a unit area, and $\mathbf{v}$ is the fluid velocity vector. The first term in the flux is diffusion, which acts to reduce concentration gradients, and the second term is motion of the chemical with the fluid.

We suppose that the source produces the chemical uniformly with flux F_{source}. To evaluate high-resolution sensing capabilities, we assume that the chemical source is small, with characteristic size L_{source} as small as individual cells. Total target surface area is $2\pi R L_{\text{source}} \approx 9.4 \times 10^{-10}\,\text{m}^2$, about the same as the surface area of a single endothelial cell lining a blood vessel. The value for F_{source} in Table 8.1 corresponds to 5×10^4 molecule$s$/s from the source area as a whole. This flux was chosen to make the concentration at the source area equal to that given in Table 8.1. The background concentration is the level of the chemical when there is no injury.

Objects, such as robots, moving in the fluid alter the fluid's velocity to some extent. For simplicity, and due to the relatively small volume of the vessel occupied by the robots, we ignore these changes and treat the fluid flow as constant in time and given by Eq. (8.2). Similarly, we also treat detection as due to absorbing spheres with concentration C at the location of the center of the sphere for Eq. (8.1), assuming that the robot does not significantly alter the average concentration around it. Figure 8.2 shows the resulting steady state concentration from solving Eq. (8.4) with $\partial C/\partial t = 0$. The concentration decreases with distance from the source and the high concentration contours occur downstream of the source due to the fluid flow.

8.4.3 Control

The limited capabilities of the robots and the need to react on millisecond time scales leads to emphasizing simple controls based on local information. For chemical detection at low concentrations, the main sensor limitation is the time required for molecules to diffuse to the sensor. Thus the detections are a stochastic process. A simple decision criterion for a robot to determine whether it is near a source is if a sufficient number of detections occur in a short time interval. Specifically, a robot considers a signal detected if the robot observes at least $C_{\text{threshold}}$ counts in a measurement time interval T_{measure}. The choice of measurement time must balance having enough time to receive adequate counts, thereby reducing errors due to statistical fluctuations,

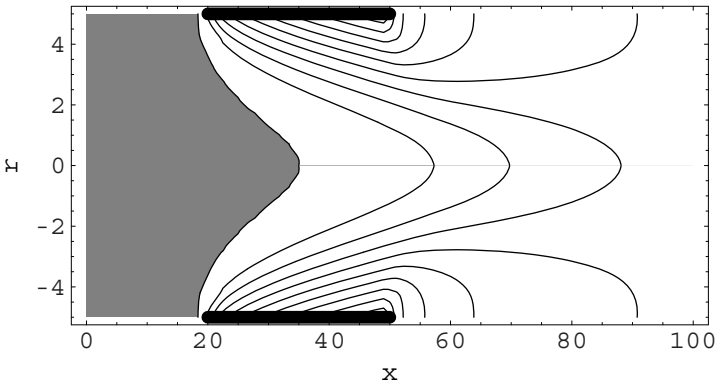


Fig. 8.2. Concentration contours on a cross section through the vessel, including the axis ($r = 0$) in the middle and the walls ($r = \pm R$) at the top and bottom. The gray area shows the region where the concentration from the target is below that of the background concentration. The thick black lines along the vessel wall mark the extent of the target region. The vertical and horizontal scales are different: the cross section is $10\,\mu$m vertically and $100\,\mu$m horizontally. Numbers along the axes denote distances in microns.

while still responding before the robot has moved far downstream of the source, where response would give poor localization of the source or, if robots are to take some action near the source, require moving upstream against the fluid flow. Moreover, far downstream of the source the concentration from the target is small so additional measurement time is less useful. A device passing through a vessel with a source will have about $L_{\text{source}}/v_{\text{avg}} = 30\,\text{ms}$ with high concentration, so a measurement time of roughly this magnitude is a reasonable choice, as selected in Table 8.2. A low value for $C_{\text{threshold}}$ will produce many false positives, whereas a high value means many robots will pass the source without detecting it (i.e., false negatives). False negatives increase the time required for detection, while false positives could lead to inappropriate subsequent activities (e.g., releasing a drug to treat the injury or infection at a location where it will not be effective).

8.4.4 Analysis of Behavior

Task performance depends on the rate robots at which detect the source as they move past it and the rate at which they incorrectly conclude that they detect the source due to background concentration of the chemical.

Table 8.2. Robot control parameters for chemical signal detection.

Parameter	Value
Measurement time	$T_{\text{measure}} = 10\,\text{ms}$
Detection threshold	$C_{\text{threshold}}$

From the values of Table 8.1, robots enter any given vessel at an average rate

$$\omega_{\rm robot} = \rho_{\rm robot}\pi R^2 v_{\rm avg} \approx 0.016/{\rm s}, \tag{8.5}$$

and the rate they enter (and leave) any of the small vessels within the tissue volume is

$$\Omega_{\rm robot} = \rho_{\rm vessel} V \omega_{\rm robot} \approx 8 \times 10^3/{\rm s}. \tag{8.6}$$

A robot encounters changing chemical concentration as it moves past the source. The expected number of counts a robot at position $\mathbf{r}$ has received from the source chemical during a prior measurement interval time $T_{\rm measure}$ is

$$K(\mathbf{r}) = 4\pi Da \int_0^{T_{\rm measure}} C(\mathbf{r}'(\mathbf{r},\tau))d\tau, \tag{8.7}$$

where $\mathbf{r}'(\mathbf{r},\tau)$ denotes the location the robot had at time τ in the past. During the time the robot passes the target, Brownian motion displacement is $\sim 0.1\ \mu$m, which is small compared to the 10-μm vessel diameter. Thus the possible past locations leading to $\mathbf{r}$ are closely clustered, and for estimating the number of molecules detected while passing the target, a reasonable approximation is that the robot moves deterministically with the fluid. In our axially symmetric geometry with fluid speed given by Eq. (8.2), positions are specified by two coordinates $\mathbf{r} = (r, x)$, so $\mathbf{r}'((r, x), \tau) = (r, x - v(r)\tau)$ when the robot moves passively with the fluid and Brownian motion is ignored. During this motion, the robot will, on average, also encounter

$$k = 4\pi DacT_{\rm measure} \tag{8.8}$$

molecules from the background concentration, not associated with the source.

With diffusive motion of the molecules, the actual number of counts is a Poisson-distributed random process. The detection probability, i.e., having at least E events when the expected number is μ, is

$$\Pr(\mu, E) = 1 - \sum_{n=0}^{E-1} {\rm Po}(\mu, n),$$

where ${\rm Po}(\mu, n) = e^{-\mu}\mu^n/n!$ is the Poisson distribution.

If the devices are taken to be ideal absorbing spheres for the chemical described in Table 8.1, Eq. (8.1) gives the capture rates $\gamma \approx 8/{\rm s}$ at the background concentration and $\gamma \approx 2300/{\rm s}$ near the source. Detection over a time interval Δt is a Poisson process with mean number of detections $\mu = \gamma\Delta t$. Consider a robot at $\mathbf{r}$. During a small time interval Δt the probability of detecting a molecule is $4\pi Da(C(\mathbf{r}) + c)\Delta t \ll 1$. For a robot to first conclude that it has detected a signal during this short time it must have $C_{\rm threshold} - 1$ counts in the prior $T_{\rm measure} - \Delta t$ time interval and then one additional count during Δt. Thus, the rate at which robots first conclude that they detect a signal is

$$4\pi Da(C + c)\frac{{\rm Po}(K + k, C_{\rm threshold} - 1)}{\sum_{n<C_{\rm threshold}} {\rm Po}(K + k, n)}. \tag{8.9}$$

In Eq. (8.9) C and K depend on robot position and the last factor is the probability that the robot has $C_{\mathrm{threshold}} - 1$ counts in its measurement time interval, given that it has not already detected the signal, i.e., the number of counts is less than $C_{\mathrm{threshold}}$. This expression is an approximation: ignoring correlations in the likelihood of detection over short time intervals. Equation (8.9) also gives the detection rate when there is no source, i.e., false positives, by setting C and K to zero.

To evaluate the rate robots detect the source as they pass it, we view the robots as having two internal control states: MONITOR and DETECT. Robots are initially in the MONITOR state, and switch to the DETECT state if they detect the chemical, i.e., have at least $C_{\mathrm{threshold}}$ counts during time T_{measure}. In the stochastic analysis approach for evaluating robot behavior, the steady state concentrations of robots monitoring for the chemical, R_{m}, is governed by Eq. (8.4) for R_{m} instead of chemical concentration, with the addition of a decay due to robots changing to the DETECT state, i.e.,

$$\nabla \cdot (D_{\mathrm{robot}} \nabla R_{\mathrm{m}} - \mathbf{v} R_{\mathrm{m}}) - \alpha_{\mathrm{m}\to\mathrm{d}} R_{\mathrm{m}} = 0. \tag{8.10}$$

The signal detection transition, $\alpha_{\mathrm{m}\to\mathrm{d}}$, is given by Eq. (8.9) and depends on the choice of threshold $C_{\mathrm{threshold}}$ and robot position.

The rate at which sensors detect the source using a threshold $C_{\mathrm{threshold}}$ is

$$\omega_{\mathrm{source}} = \omega_{\mathrm{robot}} \int \alpha_{\mathrm{m}\to\mathrm{d}} R_{\mathrm{m}} dV, \tag{8.11}$$

where ω_{robot} is given by Eq. (8.5) and the integral is over the interior volume of the vessel containing the source.

The background concentration can give false positives, i.e., occasionally producing enough counts to reach the count threshold in time T_{measure}. The background concentration extends throughout the tissue volume giving many opportunities for false positives. With the parameters of Table 8.1, the expected count from background in T_{measure} is $\mu_{\mathrm{background}} = 0.08$. Since a sensor spends $\approx L/v_{\mathrm{avg}} = 1\,\mathrm{s}$ in a small vessel in the tissue volume, the sensor has about 100 independent 10-ms opportunities to accumulate counts toward the detection threshold $C_{\mathrm{threshold}}$. The rate of false positive detections is then

$$\omega_{\mathrm{background}} \approx 100\ \Omega_{\mathrm{robot}} \Pr(\mu_{\mathrm{background}}, C_{\mathrm{threshold}}). \tag{8.12}$$

For a diagnostic task, we can pick a detection threshold $C_{\mathrm{threshold}}$ and a time T for sensors to accumulate counts. The expected number of sensors reporting detections from the source and from the background is then $\omega_{\mathrm{source}} T$ and $\omega_{\mathrm{background}} T$, respectively. The actual number is also a Poisson process, so another decision criterion for declaring a source detected is the minimum number of sensors n reporting a detection. Since the expected count rate near the source is significantly higher than the background rate, the contributions to the counts from the source and background are nearly independent, so the probability for detecting a source is

$$\Pr\left((\omega_{\mathrm{source}} + \omega_{\mathrm{background}})T, n\right),$$

and similarly for the false positives with counts based only on $\omega_{\mathrm{background}}$.

8.4.5 Detection Performance

Figure 8.3 shows the values of ω_{source} and $\omega_{\mathrm{background}}$ for the parameters of Table 8.1 for various choices of the control parameter $C_{\mathrm{threshold}}$. Despite the much larger number of opportunities for false positives compared to the single vessel with the source, the ability of robots to pass close to the source allows selecting $C_{\mathrm{threshold}} \approx 10$, for which false positive detections are small while still having a significant rate of true positives.

For diagnostics, an indication of performance is comparing the likelihood of true and false positives. In particular, identifying choices of control parameters giving both a high chance of detecting a source and a low chance of false positives. Figure 8.4 illustrates the trade-off for the task considered here. The curves range from the lower-left corner (low detection rates) with a high threshold to the upper-right corner (high detection and high false positive rate) with a low threshold. Robots collecting data for only about 20 min allow high performance, in this case with $C_{\mathrm{threshold}}$ around 10. This corresponds to the behavior seen in Fig. 8.3: $C_{\mathrm{threshold}} \approx 10$ is high enough to be rarely reached with background concentration alone (in spite of the much larger number of vessels without the source than the single vessel with the source), but still low enough that most devices passing through the single vessel with the source will detect it.

If the robots are to act at the source, ensuring at least one detection may not be enough. For instance, if the robots release a chemical near the source, or aggregate near it (e.g., to stick to a vessel wall near the source to provide enhanced imaging or to mechanically alter the tissue), then it may be necessary to ensure that a relatively large number of robots detect the source. At the same time, we wish to avoid inappropriate

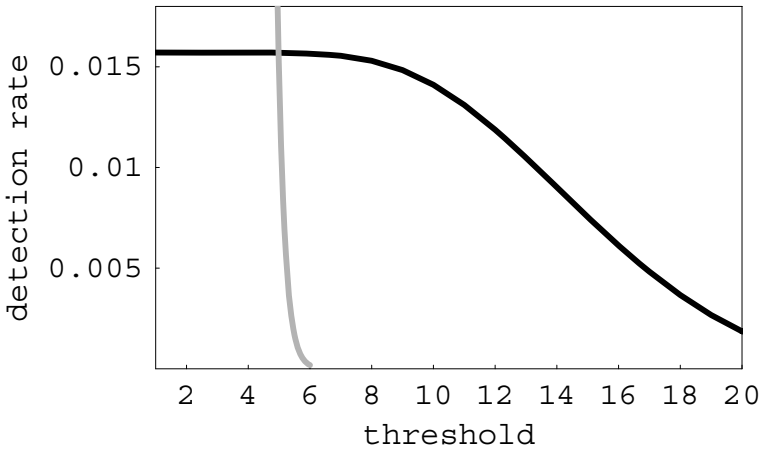


Fig. 8.3. Rates at which robots detect a signal for those passing through the single small vessel with the source (black) and all the others in the tissue volume, i.e., the false positives (gray), as a function of threshold $C_{\mathrm{threshold}}$ used for detection during the T_{measure} time interval.

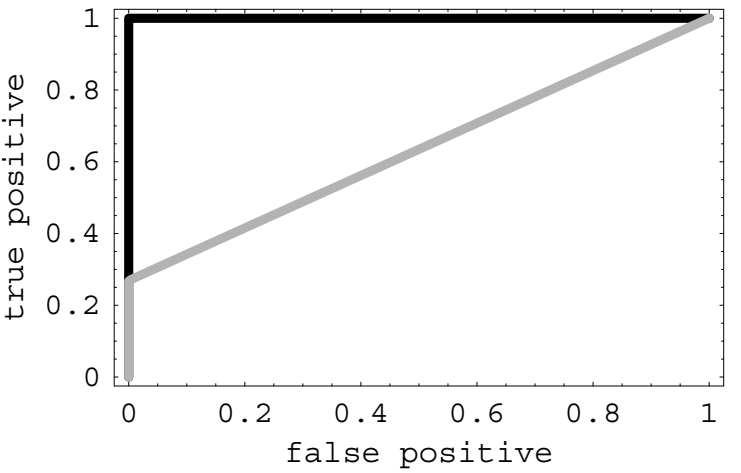


Fig. 8.4. Probabilities of detecting a single source (true positive) and mistaking background concentration for a source (false positive) after sampling for $T = 20$ and 1000 s (gray and solid curves, respectively). Each curve corresponds to changing the minimum threshold $C_{\mathrm{threshold}}$ of counts during the time interval T_{measure} required to indicate a detection event.

actions due to false positives. A criterion emphasizing safety is having a high chance that the required number detect the source before a single robot has a false positive detection, so that not even a single inappropriate action takes place. Figure 8.5 shows that we can achieve high performance using this criterion: the robots circulate for about a day to have enough time for 1000 to detect the source while still having a low chance for any false positives.

Another motivation for requiring more than one detection at the source is to account for sensor failures, e.g., requiring detection by several sensors as independent confirmation of the source. Occasional spurious extra counts by the sensors amount to an increase in the effective background concentration. As long as these extra counts are infrequent and not significantly clustered in time, such errors will not significantly affect the overall accuracy of the results. Conversely, sensor failures leading to missed counts will decrease the average detection rate, thereby requiring correspondingly longer operation times.

In summary, simple control allows fast and accurate detection of even a single cell-sized source within a macroscopic tissue volume. The key feature enabling this performance is the robots' ability to pass close to individual cells, where concentration from released chemicals is much higher than in fluid far from the cell.

For comparison, instead of using microscopic sensors, one could extract a blood sample extracted from the body, which is a routine medical procedure. Conventional laboratory analysis outside the body could then attempt to detect the chemical in the sample. However, such a sample will represent the average concentration of the chemical in the full blood volume. For a small chemical source, e.g., the size of a single cell, this average amounts to a significant dilution of the chemical, resulting in considerably

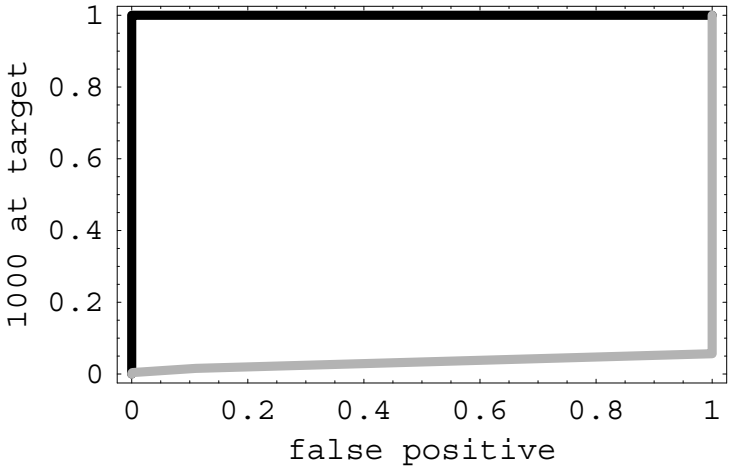


Fig. 8.5. Probability of at least 1000 robots detecting the source vs. the probability that at least one robot mistakes background concentration for a source (false positive) after sampling for $T = 6 \times 10^4$ and 10^5 s (gray and solid curves, respectively). Each curve corresponds to changing the minimum threshold $C_{\mathrm{threshold}}$ of counts during the time interval T_{measure} required to indicate a detection event.

smaller concentrations than are available to microscopic sensors passing close to the source. As an example, suppose a single source as described above produces the chemical for one day and all this production is delivered to the blood without any degrading before a sample is taken. The source producing about 5×10^4 molecule$s$/s builds up to a concentration in the 5-liter blood volume of about 7×10^{11}molecules/m^3 after a day. This concentration is about 10^{-4} of the background concentration, so the additional chemical released by the source would be difficult to detect against small variations in background concentration.

These performance estimates also indicate behavior in other scenarios. For instance, with fewer sensors, detection times would be correspondingly longer or would only be sensitive to a larger number of sources. For instance, with 10^6 sensors, a factor of 1000 fewer than in Table 8.1, achieving the discrimination shown in Fig. 8.4 would take a thousand times longer. Alternatively, for 10^6 sensors with 1000 sources distributed randomly in the tissue volume instead of just one source, performance would be similar to that shown in the figure. The stochastic analysis approach used here could also estimate other aspects of robot performance, such as the average distance to the source if and when a passing robot detects it.

The stochastic analysis approach to evaluating robot behaviors in spatially varying fields makes various simplifying approximations to obtain ω_{source} and $\omega_{\mathrm{background}}$. These include independence of the number of detection counts in different time intervals, characterizing the robots by a continuous concentration field rather than discrete objects, and ignoring how the robots (and other objects, such as blood cells) change the fluid flow and the concentration distribution of the chemical in the fluid. In principle,

the analysis approach can readily include objects in the fluid, but the numerical solution becomes significantly more complicated. Instead of the parabolic velocity profile Eq. (8.2), fluid flow changes with time as the objects move through the vessel. Simultaneously, the object motion is determined by the drag force from the fluid. Evaluating the flow requires solving the Navier-Stokes equation, which can also include more complicated geometries for the vessels and source than treated here. Despite this additional numerical complexity, analyzing robot behaviors is formally the same.

On the other hand, history-dependent behavior of the robots, such as checking for a certain number of counts within a specified time interval, is difficult to include in the formalism, leading to the simplifying independence assumptions used here. As a check on the accuracy of these assumptions, a discrete-event simulation for the same task evaluates robot behavior without making those assumptions, but requires significantly more computation time to evaluate average behaviors. Figure 8.6 shows this comparison for the key quantities used here: the detection probabilities for true and false positives. We see a fairly close correspondence between the two methods, but with a systematic error. The correspondence preserves the key qualitative difference between the curves: many orders of magnitude difference in detection probabilities over a range of thresholds (in this case around 10 to 15) where the detection probability for true positives is fairly high. This large difference allows finding a choice of threshold and measurement time giving high detection probability with low false positives, as illustrated in Fig. 8.4.

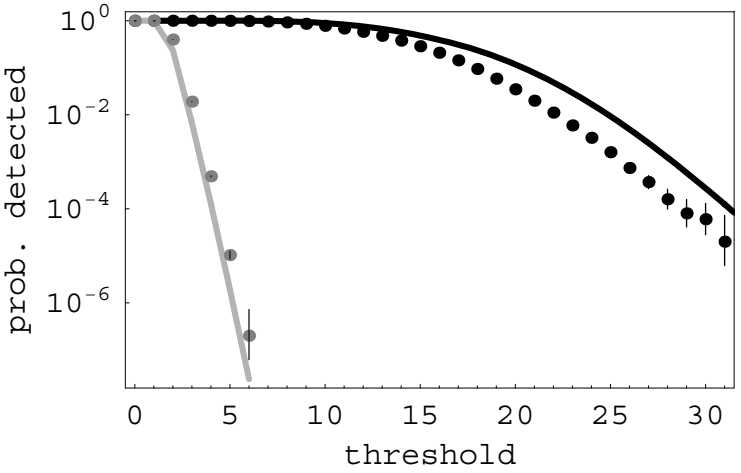


Fig. 8.6. Chemical detection probabilities vs. threshold $C_{\mathrm{threshold}}$ for detection during the T_{measure} time interval. Two cases are shown: a robot passing the source (black) and a robot in a small vessel without the source, i.e., false positive detection (gray). Curves are the theoretical estimates, and points are from simulations. Each point includes a line showing the 95% confidence interval of the probability estimate, which in most cases is smaller than the size of the plotted point. The points are averages from 10^5 and 10^7 discrete simulations of single robots passing through vessels with and without a source, respectively.

8.5 Applications for Additional Robot Capabilities

High-resolution in vivo chemical sensing with passive motion in fluids is well suited to robots with minimal capabilities. Additional hardware capabilities allow for collecting of more information communicating with external observers during operation, or taking actions such as aggregating at the chemical source, releasing chemicals, or mechanically manipulating the cells. This section presents a number of such possibilities.

8.5.1 Improved Inference from Sensor Data

The example in Fig. 8.4 uses a simple detection criterion based on a single choice of threshold $C_{\mathrm{threshold}}$. More sophisticated criteria could improve accuracy. One example is matching the temporal distribution of detections to either that expected from a uniform background or a spatially localized high-concentration source. This procedure would also be useful when the source and background concentrations are not sufficiently well known a priori to determine a suitable threshold. Instead, the distribution of counts could distinguish the low background rate generally encountered from occasional clusters of counts at substantially higher rates.

Robots could use correlations in space or time for improved sensitivity. For instance, if a source emits several chemicals together, each of which has significant background concentration from a variety of separate sources, then detecting all of them within a short period of time improves the statistical power of the inference. Similarly, if the chemical is released in bursts, sensors nearby during a burst would encounter much higher local concentrations than the time-averaged concentration. Stochastic temporal variation in protein production is related to the regulatory environment of the genes producing the protein, in particular the number of proteins made from each transcript (McAdams and Arkin 1997). Thus temporal information could identify aspects of the gene regulation within individual cells.

Correlations in the measurements could distinguish between a strong source and many weak sources throughout the tissue volume producing the chemical at the same total rate. The strong source would give high count rates for a few devices (those passing near the source), whereas multiple weak sources would have some detection in a larger fraction of the sensors.

As another example, using fluid flow sensors would allow correlating chemical detections with properties of the flow and the vessel geometry (e.g., branching and changes in vessel size or permeability to fluids).

8.5.2 Correlating Measurements from Multiple Devices

Determining properties of the chemical sources would improve if data from devices passing different sources are distinguished from multiple devices passing the same source. One possibility for correlating information from different devices is if sources, or their local environments, are sufficiently distinct chemically that measuring similar ratios of various chemicals in different devices likely indicates they encountered

the same source. Common temporal variation could also suggest data from the same source.

With less distinctive sources or insufficient time to collect enough counts to make the distinction, devices capable of communicating with others nearby could provide this correlation information directly. With the parameters of Table 8.1, typical spacing between devices is several hundred microns, which is beyond a plausible communication distance between them. Thus direct interdevice communication requires reducing the spacing with either a larger total number of devices or temporary local aggregation in small volumes.

Devices could achieve this aggregation if they can alter their surface properties to stick to the vessel wall (Lahann and Langer 2005). With such a programmable change, a device detecting an interesting event could stick when it next encounters the wall. The parameters of Table 8.1 give, on average, about one device passing through a given small vessel each minute. Thus a device on the wall could remain there for a few minutes, broadcasting its unique identifier to other devices as they pass. Devices would record the identifiers they receive with a time stamp, thereby correlating their detections.

Further inferences could be made from devices temporarily on the vessel wall where small vessels merge into larger ones. In this case, the identifier received from a device on the wall enables correlating events in nearby vessels, i.e., those that merge into the same larger vessel. Similarly, if robots aggregate in upstream vessels before they branch into smaller ones, the robots could record each others' unique identifiers. Subsequent measurements over the next few hundred milliseconds would be known to arise from the same region, i.e., either the same vessel or nearby ones.

The ability to selectively stick to vessel walls would allow another mode of operation: the devices could be injected in larger blood vessels leading into a macroscopic tissue volume of interest and then stick to the vessel walls after various intervals of time or when specific chemicals are detected. After collecting data, the devices would release from the wall for later retrieval.

8.5.3 Reporting during Operation

The devices could carry nanoscale structures with high response to external signals. Such structures could respond to light of particular wavelengths when near the skin (Wang et al. 2005) or give enhanced imaging via MRI or ultrasound (Liu et al. 2006). Such visualization mechanisms combined with a selective ability to stick to vessel walls allows detecting aggregations of devices at specified locations near the surface of the body (Service 2005).

This visualization technique could be useful even if the tissue volume of interest is too deep to image effectively at high resolution. In particular, robots could use various areas near the skin (e.g., marked with various light or ultrasound frequencies) at centimeter scales as readout regions during operation. Devices that have detected the chemicals could aggregate at the corresponding readout location, which would then be visible externally. Devices could choose how long to remain at the aggregation points

based on how high a concentration of the chemical pattern they detected. This indication of whether and (at a coarse level) what the devices have found could help decide how long to continue circulating to improve statistics for weak chemical signatures. These aggregation points could also be used to signal to the devices, e.g., instructing them to select among several modes of operation.

8.5.4 Detection of Chemicals inside Cells

The task described above relies on detecting chemicals released into the bloodstream. However, some chemicals of interest may remain inside cells, or if released be unable to get into the bloodstream. In that case, sensors in the bloodstream would not detect the chemical even when they pass through vessels near the cells.

However, an extension of the protocol could allow indirect detection of intracellular chemicals. For example, current technology can create molecules capable of entering cells and, if they encounter specific chemicals, changing properties to emit signals or greatly enhance response to external imaging methods. Such molecules can indicate a variety of chemical behaviors within cells (Xie et al. 2006). Thus if the microscopic devices include sensors for these indicator signals, they could indirectly record the activity of the corresponding intracellular chemicals in nearby cells. Such sensing from within nearby blood vessels would complement current uses of these marker molecules with much larger-scale whole-body imaging. In this extended protocol, the microscopic devices would provide the same benefits of detecting chemicals directly but by detecting a proxy signal that is able to reach the nearby blood vessels.

8.5.5 Modifying Microenvironments

Beyond the diagnostic task discussed above, robots able to locate chemically distinctive microenvironments in the body could have capabilities to modify those environments. For instance, the devices could carry specific drugs to deliver only to cells matching a prespecified chemical profile (Freitas 1999, 2006), as an extension of a recent in vitro demonstration of this capability using DNA computers (Benenson et al. 2004).

With active locomotion, after detecting the chemicals the devices could follow the chemical gradient to the source, though this would require considerable energy to move upstream against the fluid flow. Alternatively, with a sufficient number of robots so that they are close enough to communicate, a robot detecting the chemical could acoustically signal upstream devices to move toward the vessel wall. In this cooperative approach, the detecting device does not itself attempt to move to the source, but rather acts to signal others upstream from the source to search for it. These upstream devices would require little or no upstream motion. Furthermore, with a large number of devices, even if only a small fraction move in the correct direction to the source after receiving a signal, many would still reach the source. This approach of using large numbers and randomness in simple local control is analogous to that proposed for collections of larger reconfigurable robots (Rus and Vona 1999; Salemi et al. 2001; Bojinov et al. 2002). The behavior of microscopic active swimmers (Dreyfus et al.

2005) raises additional control issues to exploit the hydrodynamic interactions among swimming objects as they aggregate, so the distance between devices becomes only a few times their size (Hernandez-Ortiz et al. 2005).

Robots aggregated at chemically identified targets could perform precise microsurgery at the scale of individual cells. Since biological processes often involve activities at molecular, cell, tissue, and organ levels, such microsurgery could complement conventional surgery at larger scales. For instance, a few millimeter-scale manipulators, built from micromachine (MEMS) technology, and a population of microscopic devices could act simultaneously at tissue- and cellular-size scales. An example involving microsurgery for nerve repair with plausible biophysical parameters indicates the potential for significant improvement in both speed and accuracy compared to the larger-scale machines acting alone (Hogg and Sretavan 2005; Sretavan et al. 2005).

8.6 Discussion

Plausible capabilities for microscopic robots suggest a range of novel applications in biomedical research and medicine. Sensing and acting with micron spatial resolution and millisecond timing allows access to activities of individual cells. The large numbers of robots enable such activities simultaneously on a large population of cells in multicellular organisms. In particular, the small size of these robots allows access to tissue through blood vessels. Thus a device passes within a few tens of microns of essentially every cell in the tissue in a time ranging from tens of seconds to minutes, depending on the number of devices used. As described above, this access to cells allows rapid, chemical sensing with much higher resolution than is possible with in vitro sample analysis. Furthermore, the precision of localization and the robots' programmability gives them a degree of flexibility to alter microenvironments, e.g., by releasing drugs, well beyond that possible with either large-scale surgery or nonprogrammable chemically targeted drug delivery. The full range of biomedical situations that might benefit from this flexibility, e.g., nerve repair (Hogg and Sretavan 2005), remains to be seen.

With many devices in the tissue but only a few in the proper context to perform tasks, false positives are a significant issue. In some situations, these false positives may just amount to a waste of resources (e.g., power). But in other cases, too many false positives could be more serious, e.g., leading to aggregation blocking blood vessels or an incorrect diagnosis.

The sensing task described in this chapter highlights key control principles for microscopic robots. Specifically, by considering the overall task in a series of stages, the person deploying the robots remains in the decision loop, especially for the key decision as to whether to proceed with manipulation (e.g., release a drug) based on diagnostic information reported by the devices. Information retrieved during treatment can also indicate how well the procedure is proceeding and provide high-resolution documentation of what was done to help improve future treatments. More generally, this hybrid control illustrates an important approach to using self-organized systems: use local, distributed control to achieve robust self-organized behaviors on small scales

in space and time combined with feedback from a slower, larger central control (e.g., a person) to verify performance and consider global constraints not easily incorporated within local controllers.

The performance estimates for the sensing task show that devices with limited capabilities—specifically, without locomotion or communication with other devices—can nevertheless rapidly detect chemical sources as small as a single cell. The devices use their small size and large numbers to allow at least a few to get close to the source, where concentration is much higher than background. This chapter also illustrates the use of an analysis technique for average behavior of microscopic robots that readily incorporates spatially variable fields in the environment. Such fields are of major significance for microscopic robots, in contrast to their usual limited importance for larger robots.

As a caveat on the the results, the model examined here treats the location of the chemical source as independent of the properties and flow rates in the vessel containing the source. Systematic variation in the density and organization of the vessels will increase the variation in detected values. For instance, the tissue could have correlations between vessel density and the chemical sources (e.g., if those chemicals enhance or inhibit growth of new vessels). Accurate inference requires models of how the chemicals move through the tissue to nearby blood vessels. Chemicals could react after release from the source to change the concentrations with distance from the source. Furthermore, other chemicals, unrelated to the desired detection task, may have similar sensor binding characteristics as the desired chemical. These other chemicals will increase the effective background rate, depending on the selectivity of the sensor. Nevertheless, the simple model discussed here indicates that the devices could have high discrimination for sources as small as single cells. Thus even with some unmodeled sources of variability, good performance could still be achieved by extending the sensing time or using more sophisticated inference methods. Moreover, with some localization during operation, the devices themselves could estimate some of this variation (e.g., changes in density of vessels in different tissue regions), and these estimates could improve the inference instead of relying on average or estimated values for the tissue structure.

Further open questions include the effect of higher diffusion from mixing due to the motion of cells in fluid, for both chemicals and robots. For instance, the hydrodynamic effect of blood cells moving in the fluid greatly increases the diffusion coefficient of smaller objects in the fluid, to about $10^{-9}\,\mathrm{m}^2/\mathrm{s}$ (Keller 1971).

Instead of flowing with fluid, the sensors could be implanted at specific locations of interest to collect data in their local environments, and later retrieved. This approach does not take full advantage of the sensor size: it could be difficult to identify interesting locations at cell-size resolutions and implant the devices accurately. Nevertheless, such implants could be useful by providing local signals to indicate regions of interest to other sensors passing nearby in a moving fluid.

Safety is important for medical applications of microscopic robots. Thus, evaluating a control protocol should consider its accuracy allowing for errors, failures of individual devices, and variations in environmental parameters. For the simple distributed sensing discussed in this chapter, statistical aggregation of many devices'

measurements provides robustness against these variations, a technique recently illustrated using DNA computing to respond to patterns of chemicals (Benenson et al. 2004). Furthermore, the devices must be compatible with their biological environment (Nel et al. 2006) for enough time to complete their task. Appropriately engineered surface coatings and structures should allow sufficient biocompatibility during robot operation (Freitas 2003; Schrand et al. 2007). However, even if individual devices are inert, too many in the circulation would be harmful. From Table 8.1, sensors occupy a fraction $(4/3)\pi a^3 \rho_{\mathrm{robot}} \approx 10^{-6}$ of the volume inside the vessels. This value is well below the fraction, about 10^{-3}, of micron-size particles experimentally demonstrated to be safely tolerated in the circulatory system of at least some mammals (Freitas 2003). Thus the number of robots used in the protocol of this chapter is unlikely to be a safety issue.

Despite the simplifications used to model sensor behavior, the estimates obtained in this chapter with plausible biophysical parameters show that high-resolution sensing is possible with passive device motion in the circulatory system, even without communication capabilities. Thus relatively modest hardware capabilities could provide useful in vivo sensing capabilities far more flexible and specific than larger-scale devices. Studies of tissue microenvironments with such robots will improve knowledge of their biophysical parameters and, hence, enable better inferences from the data they collect. The improved understanding will, in turn, indicate distributed controls suitable for more capable devices and appropriate trade-offs between scale and capability for hybrid systems combining coarse centralized control with the flexibility of self-organization within the biological microenvironments.

Acknowledgments

I have benefited from discussions with Philip J. Kuekes and David Sretavan.

References

Alon, U. (2007). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman and Hall, London.

Andrianantoandro, E., Basu, S., Karig, D. K., and Weiss, R. (2006). Synthetic biology: New engineering rules for an emerging discipline. *Molecular Systems Biology*, 2(msb4100073):E1–E14.

Arbuckle, D., and Requicha, A. A. G. (2004). Active self-assembly. In *Proceedings of the IEEE International Conference on Robotics and Automation*, New York, pages 896–901.

Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., and Shapiro, E. (2004). An autonomous molecular computer for logical control of gene expression. *Nature*, 429:423–429.

Berg, H. C. (1993). *Random Walks in Biology*, 2nd edition. Princeton University Press, Princeton, NJ.

Berg, H. C., and Purcell, E. M. (1977). Physics of chemoreception. *Biophysical Journal*, 20:193–219.

Berna, J. et al. (2005). Macroscopic transport by synthetic molecular machines. *Nature Materials*, 4:704–710.

Bojinov, H., Casal, A., and Hogg, T. (2002). Multiagent control of modular self-reconfigurable robots. *Artificial Intelligence*, 142:99–120. Available as arxiv.org preprint cs.RO/0006030.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Oxford.

Casal, A., Hogg, T., and Cavalcanti, A. (2003). Nanorobots as cellular assistants in inflammatory responses. In Shapiro, J., editor, *Proceedings of the 2003 Stanford Biomedical Computation Symposium (BCATS2003)*, Stanford, CA, page 62. Available at http://bcats.stanford.edu.

Cavalcanti, A. and Freitas Jr., R. A. (2002). Autonomous multi-robot sensor-based cooperation for nanomedicine. *International Journal of Nonlinear Sciences and Numerical Simulation*, 3:743–746.

Collier, C. P., et al. (1999). Electronically configurable molecular-based logic gates. *Science*, 285:391–394.

Craighead, H. G. (2000). Nanoelectromechanical systems. *Science*, 290:1532–1535.

Dhariwal, A., Sukhatme, G. S., and Requicha, A. A. G. (2004). Bacterium-inspired robots for environmental monitoring. In *Proceedings of the IEEE International Conference on Robotics and Automation*, New York, pages 1436–1443.

Drexler, K. E. (1992). *Nanosystems: Molecular Machinery, Manufacturing, and Computation*. Wiley, New York.

Dreyfus, R. et al. (2005). Microscopic artificial swimmers. *Nature*, 437:862–865.

Freitas Jr., R. A. (1999). *Nanomedicine*, volume I: Basic Capabilities. Landes Bioscience, Georgetown, TX. Available at www.nanomedicine.com/NMI.htm.

Freitas Jr., R. A. (2003). *Nanomedicine*, volume IIA: Biocompatibility. Landes Bioscience, Georgetown, TX. Available at www.nanomedicine.com/NMIIA.htm.

Freitas Jr., R. A. (2006). Pharmacytes: An ideal vehicle for targeted drug delivery. *Journal of Nanoscience and Nanotechnology*, 6:2769–2775.

Fritz, J. et al. (2000). Translating biomolecular recognition into nanomechanics. *Science*, 288:316–318.

Fung, Y. C. (1997). *Biomechanics: Circulation*, 2nd edition. Springer, New York.

Galstyan, A., Hogg, T., and Lerman, K. (2005). Modeling and mathematical analysis of swarms of microscopic robots. In Arabshahi, P., and Martinoli, A., editors, *Proceedings of the IEEE Swarm Intelligence Symposium (SIS2005)*, New York, pages 201–208.

Gazi, V., and Passino, K. M. (2004). Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man and Cybernetics*, B34:539–557.

Ghosh, S., et al. (2003). Carbon nanotube flow sensors. *Science*, 299:1042–1044.

Gourley, P. L., et al. (2005). Ultrafast nanolaser flow device for detecting cancer in single cells. *Biomedical Microdevices*, 7:331–339.

Griffith, S., Goldwater, D., and Jacobson, J. M. (2005). Robotics: Self-replication from random parts. *Nature*, 437:636.

Hamad-Schifferli, K., et al. (2002). Remote electronic control of DNA hybridization through inductive coupling to an attached metal nanocrystal antenna. *Nature*, 415:152–155.

Hernandez-Ortiz, J. P., Stoltz, C. G., and Graham, M. D. (2005). Transport and collective dynamics in suspensions of confined swimming particles. *Physical Review Letters*, 95:204501.

Hogg, T. (2006). Coordinating microscopic robots in viscous fluids. *Autonomous Agents and Multi-Agent Systems*, 14(3):271–305.

Hogg, T., and Huberman, B. A. (2004). Dynamics of large autonomous computational systems. In Tumer, K., and Wolpert, D., editors, *Collectives and the Design of Complex Systems*, pages 295–315. Springer, New York.

Hogg, T., and Kuekes, P. J. (2006). Mobile microscopic sensors for high-resolution in vivo diagnostics. *Nanomedicine: Nanotechnology, Biology, and Medicine*, 2:239–247.

Hogg, T., and Sretavan, D. W. (2005). Controlling tiny multi-scale robots for nerve repair. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI2005)*, Menlo Park, CA pages 1286–1291. AAAI Press.

Howard, J. (1997). Molecular motors: Structural adaptations to cellular functions. *Nature*, 389:561–567.

Janeway, C. A., et al. (2001). *Immunobiology: The Immune System in Health and Disease*. Garland, 5th edition, New York.

Karniadakis, G. E. M., and Beskok, A. (2002). *Micro Flows: Fundamentals and Simulation*. Springer, Berlin.

Keller, K. H. (1971). Effect of fluid shear on mass transport in flowing blood. In *Proceedings of the Federation of American Societies for Experimental Biology*, pages 1591–1599.

Keszler, B. L., Majoros, I. J., and Baker Jr., J. R. (2001). Molecular engineering in nanotechnology: Structure and composition of multifunctional devices for medical application. In *Proceedings of the Ninth Foresight Conference on Molecular Nanotechnology*, Palo Alto, CA.

Lahann, J., and Langer, R. (2005). Smart materials with dynamically controllable surfaces. *MRS Bulletin*, 30:185–188.

Lerman, K., et al. (2001). A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, 7:375–393.

Liu, J., et al. (2006). Nanoparticles as image enhancing agents for ultrasonography. *Physics in Medicine and Biology*, 51:2179–2189.

Li, Z., et al. (2005). Silicon nanowires for sequence-specific DNA sensing: Device fabrication and simulation. *Applied Physics A*, 80:1257–1263.

Mataric, M. (1992). Minimizing complexity in controlling a mobile robot population. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, New York pages 830–835.

McAdams, H. H. and Arkin, A. (1997). Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Science* USA, 94:814–819.

McCurdy, C. W. et al. (2002). Theory and modeling in nanoscience. workshop report, www.science.doe.gov/bes/reports/files/tmn_rpt.pdf, US Dept. of Energy.

Montemagno, C. and Bachand, G. (1999). Constructing nanomechanical devices powered by biomolecular motors. *Nanotechnology*, 10:225–231.

Morris, K. (2001). Macrodoctor, come meet the nanodoctors. *The Lancet*, 357:778.

Natterer, F. (2001). *The Mathematics of Computerized Tomography*. Society for Industrial and Applied Math (SIAM), Philadelphia.

Nel, A., et al. (2006). Toxic potential of materials at the nanolevel. *Science*, 311:622–627.

NIH (2003). National Institutes of Health roadmap: Nanomedicine. Available at http://nihroadmap.nih.gov/nanomedicine/index.asp.

Patolsky, F., and Lieber, C. M. (2005). Nanowire nanosensors. *Materials Today*, 8:20–28.

Purcell, E. M. (1977). Life at low Reynolds number. *American Journal of Physics*, 45:3–11.

Requicha, A. A. G. (2003). Nanorobots, NEMS and nanoassembly. *Proceedings of the IEEE*, 91:1922–1933.

Riedel, I. H., et al. (2005). A self-organized vortex array of hydrodynamically entrained sperm cells. *Science*, 309:300–303.

Rus, D., and Vona, M. (1999). Self-reconfiguration planning with compressible unit modules. In *Proceedings of the Conference on Robotics and Automation (ICRA99)*. New York, pages 2513–2520. IEEE.

Salemi, B., Shen, W.-M., and Will, P. (2001). Hormone controlled metamorphic robots. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA2001)*, New York, pages 4194–4199.

Schrand, A. M., et al. (2007). Are diamond nanoparticles cytotoxic? *Journal of Physical Chemistry B*, 111:2–7.

Service, R. F. (2005). Nanotechnology takes aim at cancer. *Science*, 310:1132–1134.

Sheehan, P. E., and Whitman, L. J. (2005). Detection limits for nanoscale biosensors. *Nano Letters*, 5(4):803–807.

Soong, R. K., et al. (2000). Powering an inorganic nanodevice with a biomolecular motor. *Science*, 290:1555–1558.

Squires, T. M., and Quake, S. R. (2005). Microfluidics: Fluid physics at the nanoliter scale. *Reviews of Modern Physics*, 77:977–1026.

Sretavan, D., Chang, W., Keller, C., and Kliot, M. (2005). Microscale surgery on axons for nerve injury treatment. *Neurosurgery*, 57(4):635–646.

Vogel, S. (1994). *Life in Moving Fluids*, 2nd edition. Princeton University Press, Princeton, NJ.

Wang, H. et al. (2005). In vitro and in vivo two-photon luminescence imaging of single gold nanorods. *Proceedings of the National Academy of Science USA*, 102:15752–15756.

Wang, S.-Y., and Williams, R. S., editors (2005). *Nanoelectronics*, volume 80. Springer. Special issue of *Applied Physics A*. New York.

Wang, Z. L., and Song, J. (2006). Piezoelectric nanogenerators based on zinc oxide nanowire arrays. *Science*, 312:242–246.

Whitesides, G. M., and Grzybowski, B. (2002). Self-assembly at all scales. *Science*, 295: 2418–2421.

Xie, X. S., Yu, J., and Yang, W. Y. (2006). Living cells as test tubes. *Science*, 312:228–230.

Part III

Self-Organizing Computation

9

Self-Organizing Digital Systems

Nicholas J. Macias and Lisa J. K. Durbeck

9.1 Introduction

Theory is at the threshold of understanding how to translate self-organizing principles and processes to human-formed systems. However, practice lags behind theory. This chapter endeavors to provide inroads into the application of self-organization principles to one aspect of electronics systems, namely, digital logic.

Digital circuitry has proliferated from the early transistor-Transistor Logic (TTL) circuits of the 1960s to the now mass markets of computers, mobile phones, T.V.s, and numerous other consumer products. The fundamental component of digital circuitry is the logic gate from which complex functions can be derived and explained with the use of digital logic. Due to its widespread use and complex applications, digital logic is arguably a good target for applying concepts from self-organizing systems.

The ultimate goal of applying self-organization concepts to digital logic is to devise theory and practice as to how digital logic can be constructed and operated as a self-organizing system. Our approach has been to devise reconfigurable logic hardware and an architecture that permits self-organizing processes, and then to begin methodically developing self-organization concepts and their translation into practice within this framework. This work requires changing the way digital logic is both designed and built, providing so-called *primitives*, or fundamental behaviors, for self-organizing systems, along with a way to build upon these primitives to conceive of, compose, and orchestrate self-organized digital logic.

To achieve an inherently self-organizing infrastructure, a number of departures from conventional digital logic design are required. These include:

- Reworking how the system is controlled by placing the control within the componentry of the system itself; i.e., if the system is composed entirely of digital logic, then its digital logic must have the ability to control digital logic, creating a new class of digital logic that is able to change dynamically and modify both itself and its neighbors.
- The need to incorporate this self-inspecting, self-modifying power into systems without again introducing the hierarchy of controller and controlled, so that they are both loosely onto each other, or interchangeable.

- The need to conceive of and develop strategies that build upon these core capabilities to reconceive system monitoring and control as a distributed process largely enacted within the confines of the system itself and composed of many simple, localized activities with significant autonomy in their ability to decide and act on local information.

The work described below further grounds this discussion of objectives and insights with concrete examples as to how these properties are included in the hardware architecture we are developing, and gives some insight by example as to how these simple foundational or underlying processes can be used to compose digital logic that is self-organizing and dynamically self-modifying.

As this is a new approach to digital logic design, it is unlikely that we have developed all the primitives necessary for every class of problem. We therefore anticipate that as we apply this work to more classes of problems, the need for further primitives is likely. We have also not yet developed the full set of useful midlevel behaviors, built from the primitives, that are likely to be necessary for self-organizing digital logic designs. However, we report here on the current state of the art and outline areas in which this work will be further applied.

Several separate aspects of digital logic production may benefit from the application of principles of self-organization, in both the structure and function of digital logic circuits. In the case of an Field Programmable Gate Array (FPGA), a self-organizing process could be used to fabricate the physical hardware; the primitive functions of the hardware could use and enable self-organization; and any logical level could do the same for the logical layer above it by supplying primitives that the upper layers can employ. In this chapter we present research done on the design of the FPGA and its low-level logic behavior to develop self-organizing primitives that can be used to structure the logical levels above it or can be invoked by those upper logical levels. We view this as foundational work toward the eventual integration of self-organizing behavior into digital logic.

Key aspects of design at the hardware architecture and low-level system structure are the data path and the control path of the computational architecture. Much of the discussion below refers to the system *control*. This should be understood to be all those processes that direct the operation of the overall system, such as those that schedule activities or processes, synchronize the actions of disparate processes, and maintain the overall system and all its subprocesses in proper working order.

System maintenance is currently done largely by people rather than processes on the machine because of the need for physical action in many cases, such as the replacement of failing memory cards and disk drives. However, in concept, the act of inspecting the equipment for failure can be integrated into the design of the processes that run on the equipment. When systems scale upward to the point that they contain 10^{17} or more logical devices, there will likely be sufficient incentive to reorganize hardware inspection as a distributed, localized process as well, on account of the unavoidably high frequency of hardware upsets. Similarly, the process of inspecting the initially constructed hardware for defects may also become tedious enough that it has similar incentive to reorganize hardware inspection as a distributed, localized process running on the hardware itself.

To invest the low-level architecture of digital logic systems with properties of self-organizing systems, it appears to be critical to change the typical computer control path into one that is based instead upon strictly local interactions, and to then conceive of the overall control path as being a complex distributed process that emerges out of many local control actions. Our work provides this new kind of control path, and we detail a number of examples of how it is used to recast control as a highly localized process. We then describe examples in which we have built up increasingly complex systems that are composed out of these small, highly localized processes. Management of the actions of processes is thus transformed from the typical centralized control of a manager that is making decisions and controlling the system outside the system itself to distributed management and control.

Self-organization may be an antidote to the fact that the complexity of controlling and managing systems has been at least proportional to the size of the system—the number of components under management. Managing 10^{18} components using traditional methodology does not appear to be a tenable proposition. A hypothesis that underlies the work presented here is that approaches to deal with the complexity of a very large system likely require at least an equally large system as the manager. Furthermore, it may be preferable that the manager be not separate but integral with the system under management, since, e.g., management decisions come down to a very large number of small details that may be extremely difficult to output every nanosecond. Achieving this cofunctioning of manager and process under management appears possible with a particular type of infrastructure for the underlying system, one that combines sufficient flexibility with an inherently self-referential structure that makes introspection and autonomous self-modification feasible. The architecture presented here has the necessary properties to test this hypothesis. Section 9.2 describes a particular system called the Cell Matrix (Macias 1999; Durbeck and Macias 2001d) that possesses these necessary characteristics.

9.1.1 Background on the Concept of Self-Organization

The concept of self-organization has application to a number of domains. In the domain of living systems, self-organization is a central theme in many areas (Darwin 1859; Kauffman 1993). Philosophical considerations date fairly for back as well (Haldane 1931; Lennox 2001). In the domain of artificial systems, many facets of self-organization have been explored, including autonomous behavior and self-repair (Aspray and Burks 1987).

Approaches to self-organization can be grouped into at least two main categories: one we call a *statistical* approach and the other an *engineered* approach. Statistical approaches seek to manage complexity by discovering algorithms or techniques. Such approaches are not necessarily concerned with *how* the given task is accomplished, only with *how well* it is presently being accomplished. Examples of statistical approaches include neural networks (e.g., Abdi 1994) and genetic algorithms (e.g., Koza 1992). Genetic algorithms have been extensively applied to the development of electronic circuits, in a field called evolvable hardware (e.g., Thompson 1996). It should be noted that much of this work draws inspiration from biology (Darwin 1859).

In contrast to statistical approaches, engineered approaches seek to more deliberately achieve some set of goals by following more of a predefined algorithm. Interestingly, many engineered approaches also draw inspiration from biology, including the electronic embryology (embryonics) work of LSL (Ortega-Sanchez et al. 2000; Prodan et al. 2003), and the supercell work of Cell Matrix Corporation (Macias and Durbeck 2004). The discussion in this chapter falls into the domain of the engineered approach.

9.1.2 Chapter Organization

The remainder of this chapter is organized as follows: Section 9.2 describes in detail a particular target-processing architecture called the Cell Matrix, which possesses an inherently self-organizing infrastructure; Section 9.3 describes simple examples of self-modifying circuitry on the Cell Matrix, which will form the building blocks for larger-scale self-organizing systems; Section 9.4 discusses such larger systems; and Section 9.5 concludes with discussions of implementation details, including manufacturing and CAD issues related to the Cell Matrix.

9.2 Target Platform: The Cell Matrix

This chapter discusses the distributed management and control of electronic circuitry implemented on a specific reconfigurable platform called the Cell Matrix (Macias 1999; Durbeck and Macias 2001d). While there are a number of commercially available reconfigurable devices FPGAs, including many from Xilinx, Inc., most of them are essentially externally controlled; i.e., they require intervention from an outside system (e.g., a PC) in order to be configured or reconfigured (Xilinx, Inc. 2006). Moreover, even among devices that can hold several simultaneous configurations (Trimberger 1998), those configurations are generally precreated, externally, again using a PC or other extra-FPGA system. In contrast, the Cell Matrix is fundamentally an *internally configured* device: the configuration of each cell is written—and read—by the cells connected to it, i.e., its immediate adjacent neighbors. Only cells situated on the perimeter of the matrix (which are thus missing one or more neighbors) are accessible from outside the system. This is fundamentally different from most other devices, where typically every cell can be accessed from outside the system by simply sending a long configuration string throughout the device.

While it may seem unusual, and perhaps disadvantageous, to have such limited access to cells from outside the system, this is in fact a critical characteristic of the Cell Matrix and is directly linked to its ability to implement autonomous, self-organizing circuitry. Having local-only cell control also allows the system to scale without specific regard for scaling the control structures.

9.2.1 Basic Cell Structure

A Cell Matrix is a regularly tiled collection of simple reconfigurable elements called *cells*. These cells are arranged in a fixed, identical topology throughout the matrix, and

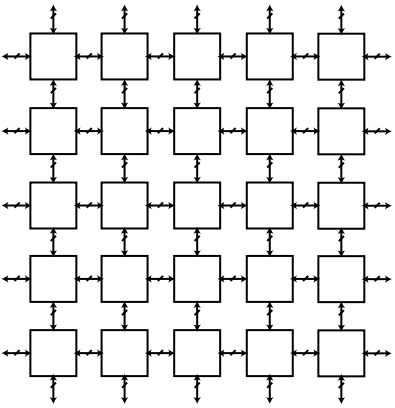


Fig. 9.1. A 5×5 collection of two-dimensional, four-sided Cell Matrix cells.

that topology defines a notion of a cell's *neighbors*: the neighbors of a cell 'X' are all those cells that are immediately connected to X. Each cell receives a single input bit (called its 'D input') from each of its neighbors and generates a single output bit (its 'D Output') to each of those neighbors. Figure 9.1 shows a two-dimensional collection of four-sided cells. In this topology, each cell has four immediate neighbors. We discuss mainly two-dimensional, four-sided cells in this chapter. However, (useful) two-dimensional cells can have as few as three sides or more than four, though four is the most typical number. Cells can also be three-dimensional, having as few as four sides, but more typically six. Higher-dimensional cells are also possible, though anything higher than three dimensions ceases to be (architecturally) infinitely scalable because there is no way to organize the cells topologically that puts all neighbors a finite, very small distance from each other.

9.2.2 Cell Structure

Each cell contains a small memory that stores a *truth table*, which maps input combinations to outputs. Given the set of incoming bits from all of a cell's neighbors, the cell's outputs are precisely determined by the information in the cell's truth table. This mechanism allows a single cell to implement simple combinatorial functions, such as basic logic gates, single-bit adders, or multiplexers. Cells can also act as simple wire: a block for passing data from one side of itself to another—or, viewed differently, a block for allowing two nonadjacent cells to share data with each other. Implementing wires is a major use of cells.

9.2.3 Cell Configuration

The act of loading truth table information into a cell is called *cell configuration* or *configuring a cell*. Similarly, the act of loading truth table information into a number

of cells is called *configuring the Cell Matrix*. While single-cell circuits are necessarily extremely simplistic, configuring a group of cells appropriately leads to multicell circuits, which can be arbitrarily complex. As single cells can implement any fixed input-to-output mapping and cells can be interconnected via intervening cells, any circuitry that can be implemented using traditional digital circuit design can also be implemented on a Cell Matrix.

Figure 9.2 shows a more detailed view of a single cell. As can be seen, each side has two input lines and two output lines, which connect it to each of its immediately adjacent neighbors. One line is labeled D and the other C. The D inputs are used to select information from the cell's truth table, and the D outputs are set based on the truth table's values, as described above. *But this is the case only if all the C inputs are 0.*

When all C inputs are 0, the cell is said to be in D mode. If, however, any C inputs are set to 1, then the cell is in C mode, which is the configuration mode of a cell and the mode in which a cell's truth table can be modified. In C mode, a cell's truth table can also be examined. A cell that is asserting one of its own C outputs, and is thus asserting a neighboring cell's C input, is able to read and write that neighboring cell's truth table. Note that if more than one C input is set to 1, then the cell's truth table is sent to multiple neighbors, and its new truth table is determined by a combination (logical ORing) of its neighbors' outputs.

This C-mode operation is like the *unit measure* of self-organization for the entire architecture; using it, cells are configured by a neighboring cell. The extreme locality of this operation makes the entire architecture fine-grained in its reconfigurability, and makes configuration a distributed, local process.

Cell configuration is the only inherently clocked operation in the Cell Matrix: a single system-wide clock is used to serially shift out the current contents of a cell's truth table and into new truth table bits. These bits are read from and written to the D output and input lines, respectively, on the same side on which the cell's C input is asserted (called the *active side*). Figure 9.3 shows an example of adjacent-cell interactions in D and C modes. In Fig. 9.3a, cell Y is in D mode, since all of its C inputs are 0. It thus uses its four D inputs to select a single row in the 16×8 truth table memory and sends the selected eight output values to its eight outputs (four C and four D).

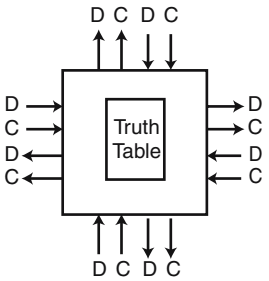


Fig. 9.2. Four-sided cell matrix cells. Each neighboring cell reads and writes two bits (D and C). C inputs determine the *mode* of the cell. D inputs either select C and D output values from the truth table (in D mode) or supply new values for the truth table (in C mode).

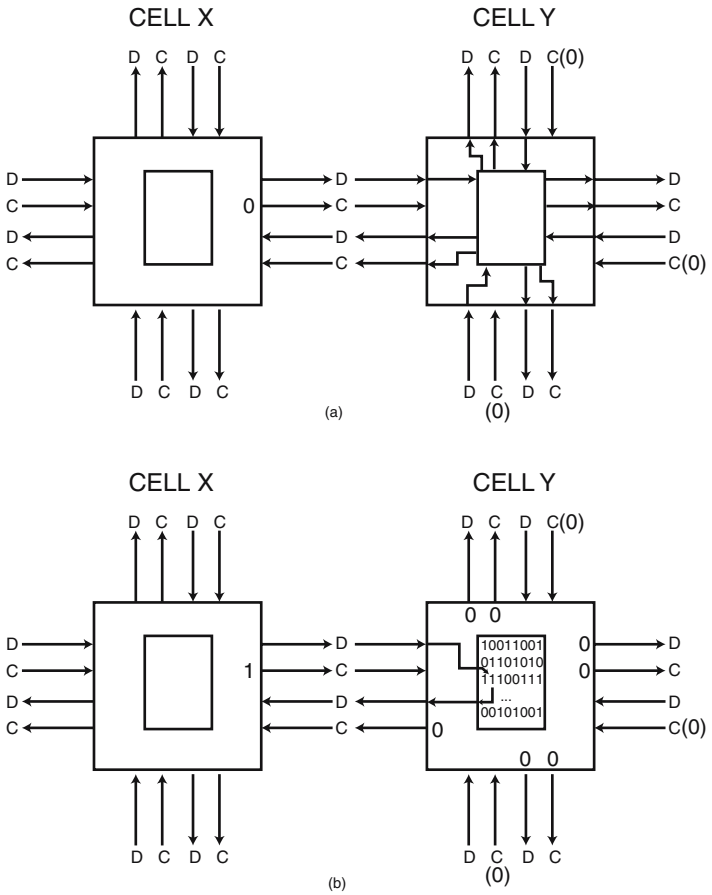


Fig. 9.3. The two modes of cell operation. (a) Cell Y is in D mode (all C inputs are 0). Its four incoming D values are used to select eight output values from its truth table. Those output values are sent to the cell's eight output lines (four C lines and four D lines). (b) Cell X is asserting a 1 to one of cell Y's C inputs, and thus cell Y is in C mode. In this mode, the D input from cell X supplies new values for Cell Y's truth table. Each time the system clock ticks, a new incoming bit value is sampled and loaded into cell Y's truth table. Cell Y's current truth table bits are simultaneously sent out the D output to cell X. The second bit in the third row of the truth table is being read and written by cell X. All of cell Y's other outputs (C and D) are forced to 0.

In Fig. 9.3b, one of cell Y's C inputs is asserted (the one supplied by cell X). This places cell Y into C mode, the mode in which its truth table is read and written. Each time the system-wide clock ticks, the D input supplied by cell X is loaded into cell Y's truth table, at a position that changes with each tick (in Fig. 9.3b, the bit in the third row, second column is being written). Additionally, the previous value stored in that location is made available on the D output to cell X. All other D outputs from cell Y are forced to 0, as are *all* of cell Y's C outputs (so that a cell being configured cannot

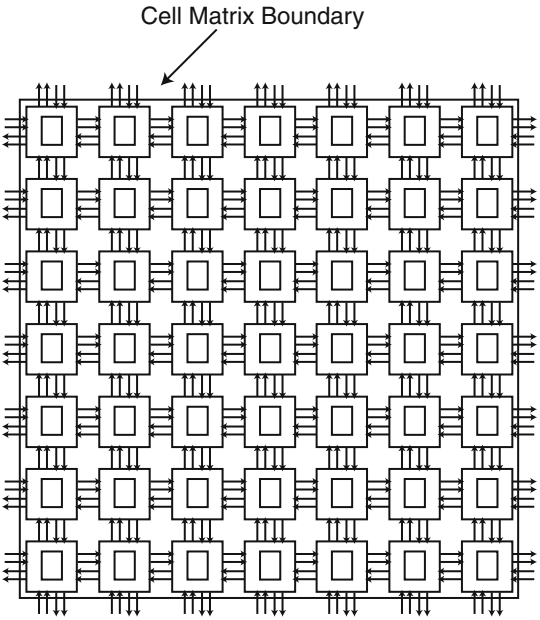


Fig. 9.4. A 7×7 Cell Matrix. Edge cells have their D and C inputs and outputs accessible from outside the matrix. Corner cells have I/O accessible on two of their sides.

itself simultaneously configure another cell). By convention, if two or more of a cell's C inputs are asserted, the bit value loaded into the cell's truth table is the logical OR of the D inputs on all active sides.

With this simple interaction scheme, it is possible for any cell to read and write any neighboring cell's truth table. Since cells along the edge of the matrix have some of their inputs and outputs unconnected, as shown in Fig. 9.4, those edge cells can be configured from outside the matrix, if their C and D inputs (and, perhaps, outputs) are made available. In Fig. 9.4, all edge cells have their inputs and outputs accessible from the edge of the matrix on at least one cell side (corner cells are accessible from two sides).

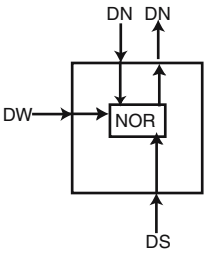


Fig. 9.5. A single cell implementing a three-input NOR gate.

INPUTS				OUTPUTS							
DN	DS	DW	DE	CN	CS	CW	CE	DN	DS	DW	DE
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0

Fig. 9.6. Truth table corresponding to Fig. 9.5.

Figures 9.5–9.7 illustrate the full details of a cell configuration operation: Fig. 9.5 shows a single cell configured as a NOR gate; Fig. 9.6 shows the cell's corresponding truth table; and Fig. 9.7 shows a timing diagram for configuring the cell. The cell is first placed into C mode by raising one of its C inputs. As soon as it enters C mode, the current value of the cell's first truth table bit is sent to the corresponding D output (not shown in the figure). On the next rising edge of the system clock, the D input is sampled and latched. On the next falling edge, the latched value is loaded into the truth table, and the current truth table's next bit is sent to the D output. Note that this timing

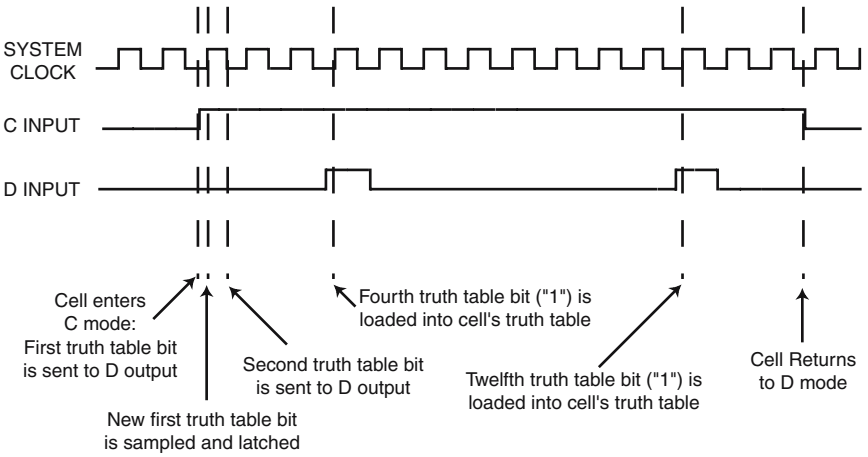


Fig. 9.7. Truth table programming sequence. After the cell is placed into C mode, a 1 bit is loaded on the 4th and 12th ticks of the system clock. The cell is returned to D mode two ticks later. This loads 14 bits into the cell's truth table: 0001 0000 0001 00.

makes the truth table's current bit values available on the D output half a cycle before the new bit value must be presented to the D input. This makes it simple to read a truth table bit and then rewrite the same bit, thus performing a nondestructive read.

Before the fourth clock tick, the D input is raised. This 1 value is latched/loaded into the cell's truth table on the next rising/falling edge of the system clock. Similarly, a 1 is loaded during the 12th cycle following the cell's entry into C mode. All other incoming bit values are 0.

A few cycles later, the cell's C input is set to 0, and the cell returns to D mode. If its truth table initially contained all 0s, it is now as shown in Fig. 9.6.

9.2.4 Self-Configuration

As cells are able to read and write other cells' truth tables, the Cell Matrix can be configured from *inside* itself, which makes it a *self-configurable* system; i.e., circuits can be constructed that read and write cell configurations, and thus they can analyze and change circuitry within the matrix. Circuitry constructed on the Cell Matrix can process data that represents logical values, characters, integers, floating point numbers, or any sort of data structure. But additionally, circuitry constructed on the Cell Matrix can also process a unique type of data: circuit configuration information. Furthermore, because the mapping between circuit configuration and circuit behavior is very straightforward, one can construct circuits that effectively *process other circuits*.

Moreover, there is no hardware-level or architectural difference between a cell that is being configured and one that is configuring it. Figure 9.8 shows some of the possibilities resulting from this fact. In Fig. 9.8a, cell X, cell Y, and cell Z are all in D mode, since their C inputs are all 0 (all inputs are assumed to be 0 unless shown otherwise). Each cell is simply receiving D inputs, using them to address their internal truth table, and producing D and C outputs accordingly.

In Fig. 9.8b, cell X is asserting a 1 on its C output to cell Y, which places cell Y into C mode. In this mode, cell Y's truth table is being configured by cell X by sampling the D outputs sent from cell X to cell Y.

In Fig. 9.8c, cell X has returned its C output to 0, and thus cell Y returns to D mode. Cell Y is thus asserting its outputs based on the (new) contents of its truth table. In this example, the truth table indicates that cell Y's C output to cell Z is to be set to 1. This places cell Z into C mode, and cell Y is now configuring cell Z.

This example illustrates a very typical case: cell Y was previously configured by a neighbor, but it is now itself configuring another neighbor. Within the Cell Matrix, there is perfect interchangeability between subjects and objects of configuration operations, which is the essence of self-configuration and self-modification within the Cell Matrix.

9.2.5 Implications

There are a number of immediate implications arising from the architecture described above. The Cell Matrix *architecture* is infinitely scalable. Because only power and a

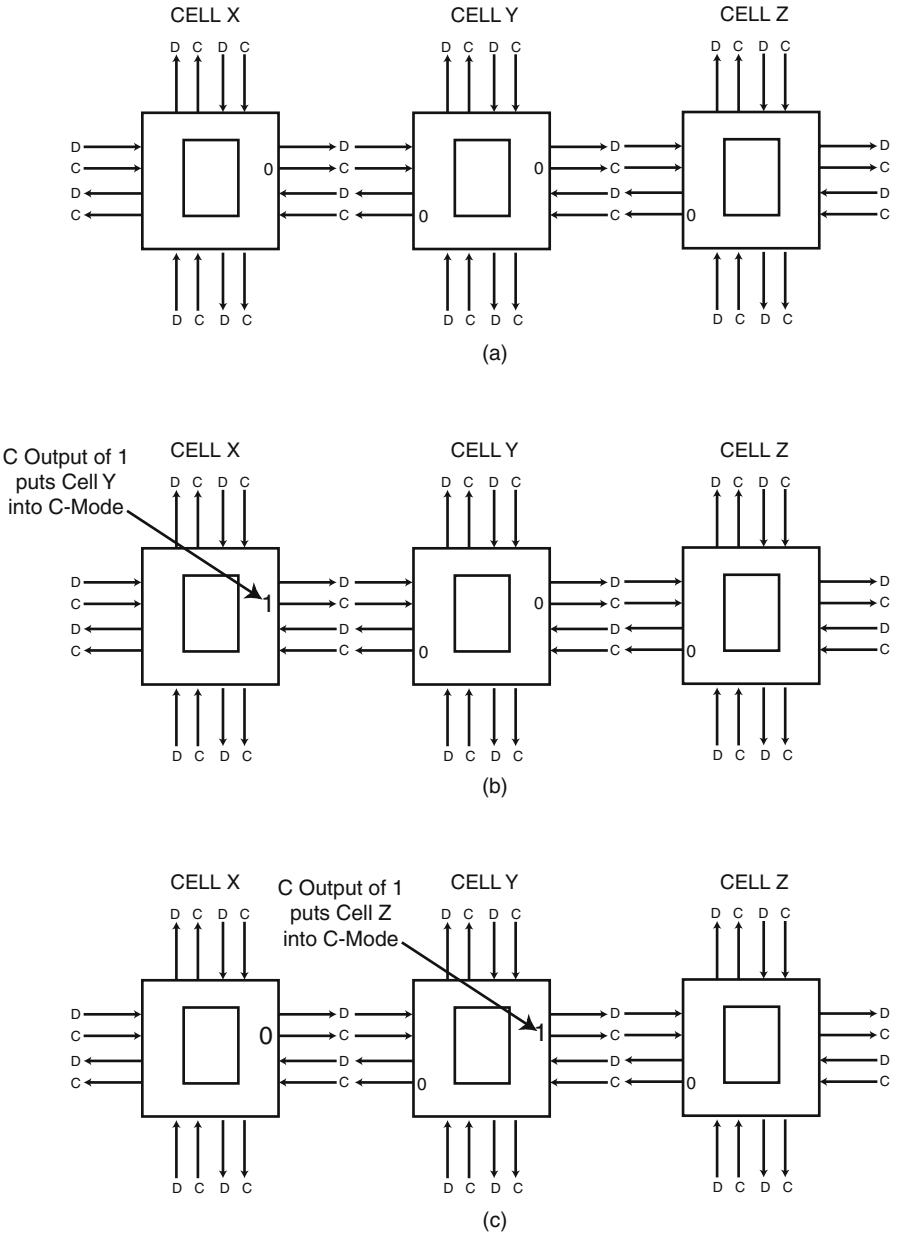


Fig. 9.8. Interaction of cells' modes. (a) All three cells are in D mode, with each cell reading inputs and producing outputs based on its current truth table contents and D inputs. (b) An input change (not shown) in cell X's D inputs has caused cell X to assert its C output to cell Y. Cell X has thus placed cell Y into C mode, and cell Y's truth table is now being configured. (c) Cell X is again outputting a 0 on its C output to cell Y; cell Y has thus been returned to D mode. Based on cell Y's new truth table, it is now asserting its C output to cell Z and has thus placed cell Z into C mode; cell Z is now being configured by cell Y.

single clock line are distributed throughout the matrix, there is no architectural impediment to scaling a matrix to whatever size is desired. Put another way (and, again, assuming a fixed dimensionality and interconnection topology), all submatrices of a given size are identical to each other, no matter what matrix they are embedded in: the structure of the cells and their interconnections is independent of the larger matrix to which they belong.

This means that two matrices can be combined into a large matrix simply by connecting the matrices to each other along an edge, i.e., connecting one matrix's edge cells' input to the other's edge cells' outputs, and vice versa. The architecture scales up without change (though of course the maximum possible latency increases). This also has interesting manufacturing implications (see Sec. 9.5).

As the configuration of cells is essentially a local operation, there is no such thing as *runtime vs. configuration time* for the matrix at large and no need to discuss *runtime reconfiguration*. The matrix is *always* running, and part of its running operation may include reconfiguration operations. Moreover, *partial configuration* (Schmit 1997) is the only type of configuration ever performed, since any single configuration operation affects only the neighbors of the cell being configured.

The Cell Matrix is completely homogeneous in structure. Cells are differentiated by their configuration information (truth table contents), but at the underlying hardware level, they are identical to one another, just as are their interconnections to other cells. This has tremendously beneficial manufacturing implications. It also has positive implications for circuit reliability, since no piece of the matrix (or the circuits implemented on top of the matrix) is unique or irreplaceable.

Owing to the capacity for self-modification in the Cell Matrix, high-level configuration mechanisms can be designed, tailored to the specifics of the target circuit, and then constructed out of cells. Moreover, the construction of the configuration mechanism *can itself be constructed* by using a previously created configuration mechanism. In this way, configuring the Cell Matrix may closely resemble a traditional *bootstrap* process, wherein a simple circuit is first built using the limited control available from the edge of the matrix. This simple circuit is then used to configure a more complex circuit, which is then used to configure a more complex circuit, and so on, building more and more complex circuits until the desired configuration has been achieved. Moreover, note that while a single set of commands may be used repeatedly to configure multiple Cell Matrix regions, it is still possible to introduce *differentiation*, including randomness, into the configured circuits.

Finally, because cells are configured by neighboring cells, it is possible for multiple cells to be configured simultaneously, either with the same or completely different configurations one from the other.

9.2.6 Status

The Cell Matrix architecture has been fully documented (Macias et al. 1999; Durbeck and Macias 2001c; Macias and Raju 2001; Cell Matrix Corporation 2006a,b). A variety of simulators and debuggers have been developed, as have various tools for developing

circuitry on the matrix. Prototype tools for converting from abstract netlists to Cell Matrix configuration information have been developed (Macias 2006).

A number of fairly traditional circuits have been implemented on top of the Cell Matrix, including state machines, arithmetic units, memories, floating point processors, and cellular automata simulators. Section 9.4 describes some of the less traditional circuits that have been implemented, including circuits that utilize self-configuration.

Furthermore, although the high-level behavior of the Cell Matrix is well-defined, it has multiple possible implementations. For example, the original implementation (Macias et al. 1999) utilized a shift register for each cell's truth table. While this simplifies the design of each cell, it means that the entire truth table changes during a configuration operation. Later work produced a slightly more complicated cell implementation that utilizes a nonshifting memory that takes up a much smaller fabrication area (Durbeck and Macias 2001c). A further modified cell incorporates bypass logic to detect when a cell is acting as a wire and to directly connect an input to an output, greatly improving signal transmission rates when cells are used as wires (Macias and Raju 2001).

9.3 Building Blocks of Self-Configuring Circuitry

This section describes some of the primitives of self-configuration for the Cell Matrix, the basic building blocks and techniques related to the implementation of self-configuring circuitry on the Cell Matrix. Section 9.4 describes higher-level circuitry constructed from these blocks.

9.3.1 Cell Replication

Figure 9.9 shows a simple cell-replication circuit that copies the truth table of the source cell into the target cell. The cell in the middle is the *controller* of the configuration operation. The cell to be replicated (called the *source cell*) is on the right. The *target cell*, which will become a copy of the source cell, is on the left. When a 1 is sent into the northern D input of the controller, it asserts its C outputs on the left and right, thus placing the source and target cells into C mode. Each cell then begins outputting current truth table bits on one of its D outputs (on the side where C_{in} is asserted), as well as receiving new truth table bits on the same side's D input. The controller reads bits from the source cell and sends them back into the source cell, thus rewriting the source cell's truth table while it is being read. Additionally, the controller sends the source cell's truth table bits into the target cell's D input, thus configuring the target cell's truth table as an exact copy of the source cell. After a sufficient number of ticks of the system clock (128 for four-sided cells, i.e., enough ticks to sample and write each of the truth table bits for a 16×8 truth table), the target cell's truth table will match the source cell's, and thus the target cell will behave exactly the same as the source cell: the source cell has effectively been replicated by the controller.

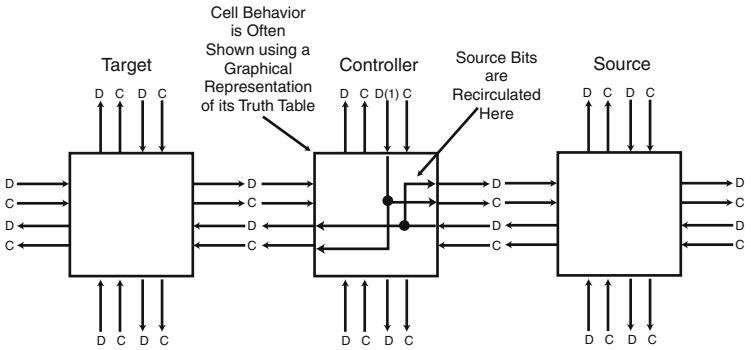


Fig. 9.9. Single-cell replicator. When a 1 is sent into the controller's northern D input, it places and source and target cells into C mode, reads truth table bits from the source, copies them back to the source, and also copies them to the target. After 128 ticks of the system clock, the target will be an exact copy of the source. Because the source cell's truth table bits are loaded back into the source cell's D input, the source cell's truth table is left unchanged by this circuit. This is thus a nondestructive read.

9.3.2 Remote Cell Replication

Figure 9.10 shows a circuit that is similar to that in Fig. 9.9, except that the source and target cells are not adjacent to the controller. Instead, they are now some distance away from the controller, and cells located in between them are used to transmit C and D information between the controller and the source and target cells. The controller works the same as in Fig. 9.9, except that it cannot directly control the mode (C or D) of the source and target cells. This makes the circuit in Fig. 9.10 somewhat limited in its usefulness. A more useful approach involves the use of *multichannel wires*.

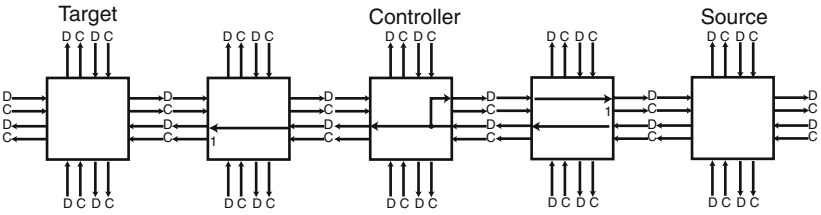


Fig. 9.10. Remote cell replicator. The source and target cells are in C mode. The source cell's truth table bits are read by the controller, sent back to the source cell, and also copied to the target cell. Note that the controller no longer directly controls the mode of the source and target cells.

9.3.3 Multichannel Wires

To control a cell, it is generally necessary and sufficient to control the C input, D input, and D output on one of the cell's sides. The circuit shown in Fig. 9.11 is an example of a structure for controlling nonadjacent cells by utilizing two lines of intervening cells. Such a structure is called a *multichannel wire*. In this circuit, the controller sends information along two lines of cells (each called a *channel*).

The bottom channel (called the C channel) controls the C input on the source and target cells, while the top channel (called the D channel) accesses the D input and output on the source and target cells. Note that these lines are logical wires, or soft wires, rather than hard, physical wires; they are created by setting the truth tables of the cells to pass their input directly to their output. This primitive gives the controller more or less complete control over the source and target cells: the ability to place them in C mode, read and write their truth tables, and then return them to D mode.

There are other types of multichannel wires, but they all have the same basic characteristic: they allow a set of cells to interact with one or more nonadjacent cells. By using the right types of multichannel wires, a set of controller cells can thus configure cells that are not adjacent and not directly connected to itself.

While it is evident from this example that wires allow access to nonadjacent cells, it would appear that they only allow access to the cell adjacent to the end of the wire, but that is not the case. If the target cell is treated as a controller, then it is possible to access cells that are near, but not adjacent to the end of the wire. For example, if the target cell (call it *X*) is configured as shown in Fig. 9.11, then data subsequently transmitted to cell *X* will, in fact, be used to configure the cell *below* cell *X*, the cell shown in Fig. 9.12 effectively moves the location of the wire's target cell.

Therefore, even though a wire can directly control only the cell adjacent to its end, it can indirectly control nonadjacent cells using intermediate cells such as that shown in Fig. 9.12.

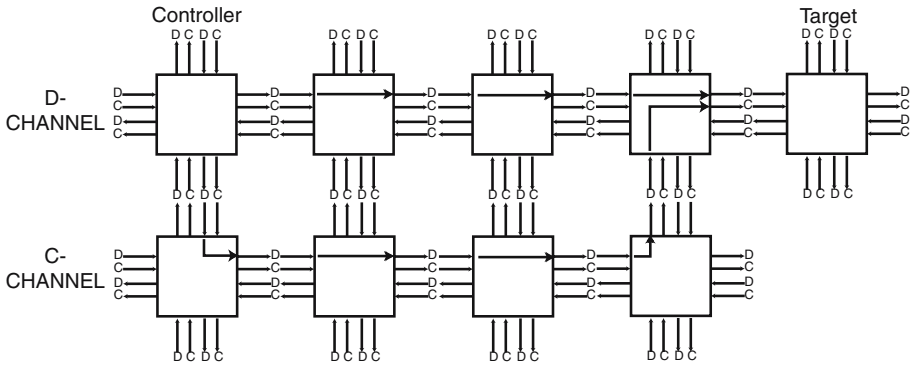


Fig. 9.11. Simple multichannel wire. The controller sends new truth table bits into the target via its own eastern data output. Additionally, the controller can now control the mode of the target via its own southern data output. This is a significant improvement over the circuit shown in Fig. 9.10.

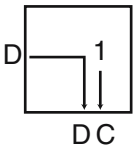


Fig. 9.12. A target cell that can be used to configure a cell near a wire's target cell.

Repeated application of this technique could, in theory, be used to gain control over a cell located anywhere within the matrix, but it is limited in its usefulness, since accessing cells n locations away requires on the order of 2^n steps. Thus, wires' only practical use during configuration is to manipulate cells near their end. This would be a severe limitation of the Cell Matrix's nearest neighbor topology, were it not for the concept of *wire building*.

9.3.4 Wire Building

Special kinds of wire building permit a cell to access any cells within a Cell Matrix. Although wires can only be used to control cells adjacent to their ends, *those wires themselves are built out of cells*. In fact, it is possible to design a wire that allows cells at its end to be configured in order to make a new piece, i.e., to extend the wire. Beginning with a short wire, cells at its end can be configured to make the wire one cell longer. That longer wire can be used to configure the cells at the new end, thus making the wire another cell longer, and so on. This process can be repeated indefinitely (as long as there are cells available), thus allowing a set of controller cells to access cells arbitrarily far away. Moreover, it is possible to create wires that have turns, and again these turns can be created by the controller. This means a set of cells can, in fact, access any cells within the matrix through the use of proper wire-building techniques. Note that this permits remote control of cells, even though the underlying primitives are all strictly neighbor to neighbor. Remote control permits external control of the system, but it also permits the smallest unit of a complex system to be multicelled to an arbitrary size, which is convenient for most applications, including most work in self-organizing systems. In Section 9.4 we describe an application that uses a supercell as its unit that contains 270×270 cells.

Figures 9.13 shows a sample two-channel wire that is *extendible*. As in the wire of Fig. 9.11, the upper channel transmits a D signal, and the lower one transmits a C signal. The cell labeled with an (*) is again called the target cell, and as in Fig. 9.11, both cell (*) and cell (**) can be easily configured. However, unlike the two-channel wire shown in Fig. 9.11, *all of the D channel cells are identical* and *all of the C channel cells are identical*. This is accomplished through the use of a feedback signal: each cell within the D channel asserts a 1 to its south, which the corresponding C channel cell transmits to the previous cell of the C channel. Therefore, the end of the wire is identified not by a differently configured cell, but rather by the lack of this feedback signal. Thus, by simply creating a new pair of D channel and C channel cells at the

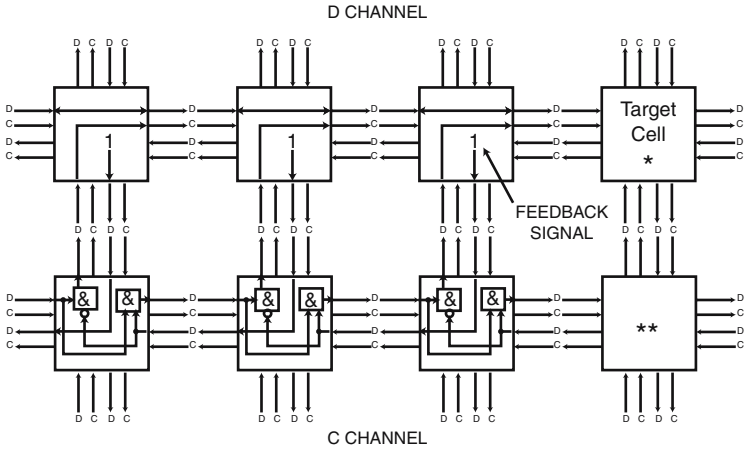


Fig. 9.13. Two-channel extendible wire. The D channel transmits configuration information for the target Cell. The C channel controls the mode of the target cell. The feedback signal is used for autonomous determination of the location of the wire's end. If cells (*) and (**) are configured as new channel pieces, then the wire will automatically be extended.

end of the wire, the wire is effectively extended, i.e., the location of the target cell is shifted one cell to the right.

This is the essence of wire building. While different sequences are needed for different types of extensions (such as turns) and for different types of wires (such as three-channel wires), the basic mechanism is the same as in Fig. 9.13. These mechanisms are described in more detail elsewhere (Macias 2001).

Multichannel wires and associated wire-building techniques can be used to access cells anywhere within the matrix, which raises the question, "What does one do with such access?" There are many answers to this, and in the remainder of this section, a few general examples are presented. Section 9.4 discusses more-specific examples.

9.3.5 Cell Testing

Given access to the C input and D inputs and outputs of a target cell, it is possible to perform a variety of tests on the target cell to ascertain its health, i.e., to determine if it is operating as expected. For example, the cell's truth table can be loaded with 0's, and then the D output examined while the D input is toggled between 0 and 1. This would detect shorts between input and output, as well as stuck-at-one faults inside the truth table memory or along the D input or output paths. A second example is that a set of certain bit patterns can be loaded into the cell, and then read back out and compared to the loaded pattern: different alternating-bit patterns can be used to detect shorts within the truth table memory based on the physical layout of the memory within the cell. This fault-testing work was developed and successfully conducted on defective hardware for the Cell Matrix architecture using the above-described multichannel wire building to reach each cell (Durbeck and Macias 2002).

9.3.6 Circuit Building

The question of how to *bootstrap* a Cell Matrix remains; i.e., with no direct access to the vast majority of cells within the Cell Matrix, how can a matrix be populated with the desired set of truth tables, particularly given that the matrix is always running, so that truth tables are in use from the moment they are in place. However, all the necessary building blocks have already been presented. Figures 9.14a–f show a sample bootstrap sequence. Note that this is not the only possible way to bootstrap a region of the Cell Matrix. It is a pedagogically interesting example because it is one of the simplest and most straightforward, but it is not used in typical practice because it is one of the slowest ways to configure a region of cells.

In Fig. 9.14a, a two-channel wire is built from west to east, extending just one column of cells shy of the easternmost corner of the region of interest. The target cell in the corner of the region is then configured. In Fig. 9.14b, the wire has made a corner and is extended one step to the south. This extended wire is then used to configure the next target cell (*). The wire is then extended south another step, and a third target cell is configured to the east, as shown in Fig. 9.14c. This process continues, until, as in Fig. 9.14d, an entire column of target cells has been configured, along the easternmost edge of the region of interest. In Figure 9.14e, the wire has been broken, i.e., the end of the wire is returned to the original entry location into the region of interest. The wire is again extended to the east, but stops one cell earlier. The wire then turns a corner, and the above steps (configure/extend) are repeated, configuring a second column of cells, as shown in Fig. 9.14f.

The above steps are repeated until the entire region of interest has been configured. Note that this technique cannot be used exactly as described for configuring the

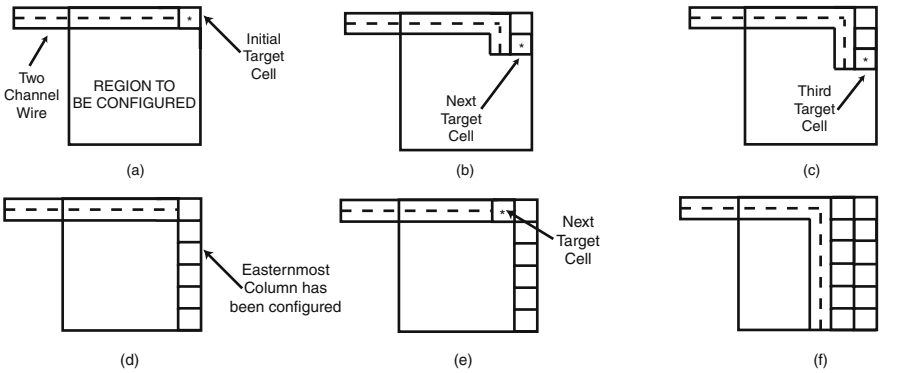


Fig. 9.14. Configuration of a region. (a) A two-channel wire is built into the region of interest and used to configure cell (∗). (b) The wire has been extended with a corner, and the next target cell is configured. (c) The wire is extended further to the south, and a third target cell is configured. (d) The easternmost column has been completed. (e) The beginning of the second column's configuration; the wire has been broken and rebuilt, but ends one cell shy of the previous extension. (f) The second column has been completely configured. This process is repeated until the entire region has been configured.

westernmost columns, since the wires themselves have a width to them. The easiest way to address this is to avoid this edge case by imagining the region of interest as being one wire width wider than it really is and leaving the westernmost columns (which are not actually of interest) unconfigured.

There are numerous enhancements to this basic scheme, including techniques to avoid completely rebuilding the west-to-east wire after each column pass. Parallel configuration is also feasible, and is discussed briefly in Section 9.4. Also, note that the configuration of cells that assert their C outputs requires special consideration, since such outputs could interfere with the configuration of the wires that are being used to configure the region's cells.

9.3.7 Circuit Reading

With circuits and sequences similar to the bootstrap method described above, it is possible to nondestructively read a set of cell configurations from a region of the matrix. The technique is similar to bootstrapping, but utilizes a *reversible* wire, i.e., one that can not only be extended a single step, but can also be shortened a single step. The basic technique is shown in Figs. 9.15a–f. For simplicity, this illustrates the reading of a single (one-dimensional) line of cells only. Note that implementation of a reversible wire requires three channels.

In Fig. 9.15a, the wire has been built to the east, to the beginning of a set of cells whose contents are to be read. The cell directly ahead of the wire (i.e., to the east of the D channel) is read, and its truth table configuration is stored in a temporary repository (a FIFO). That cell is then configured to allow reading of the cells to the north and south of it, with those cells' configurations also being stored in the temporary repository. This is shown in Figs. 9.15b and c, respectively.

The three-channel wire is then extended and the process repeated. In Fig. 9.15d, the wire has extended all the way to the east. Again, edge cases need to be considered, but can be neglected by imagining the region of interest to be larger than it actually is. At this point, all of the cells occupied by the wire have been reconfigured from their initial configuration (in order to implement the extended wire), but their initial configurations have been read (and presumably processed by some circuitry outside that shown in these figures). Moreover, *those configurations have been stored in a temporary storage location*, which can be as simple as a set of cells arranged in a two-way shift register, i.e., a FIFO.

In Fig. 9.15e, the wire is reversed a single step (using the third channel), and in Fig. 9.15f, the previously stored configurations are restored to the cells near the end of the wire. These two steps are repeated until the entire row has been restored.

For a two-dimensional region, another pass would be made to the south of the original west-east wire, thus reading the next three rows of cells. Eventually, the cells within some region of interest will all have their initial configurations, but a copy of those configurations will have been sent by this circuit to some other circuitry that will perform an analysis, make a new copy, vote on truth table contents among multiple copies, or conduct some other function.

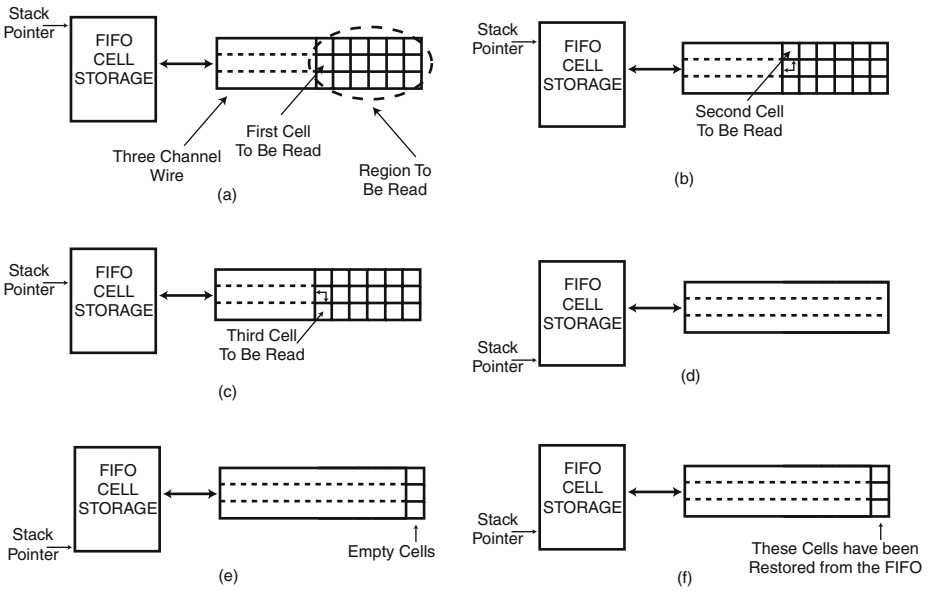


Fig. 9.15. Nondestructive read of a region of cells. (a) A three-channel wire has been built to the edge of a region to be read; a first cell is read and its configuration is stored in the FIFO. (b) That first cell is used to read a second cell, which is also stored in the FIFO. (c) A third cell is read and stored, after which the wire is extended one step. (d) The entire region has been read, stored, and overwritten with the wire itself. (e) The wire is reversed (backed up one step). (f) The easternmost cells have been restored form the FIFO.

Note, however, that while the above technique will read the configuration of cells without (permanently) changing them, it does *not* read the state of cells, i.e., the values of their inputs and outputs, and similarly does not preserve their state. State reading and preservation would require additional circuitry built into the circuit itself, since changing the configuration of a single cell can, in general, alter the state of the entire circuit.

These are a few detailed examples of techniques related to self-configuring circuitry. They form a base of primitives or building blocks that are composed to create more complex functions and circuits. The next section describes larger-scale applications of these techniques to the implementation of circuits that exhibit distributed management and control.

9.4 Distributed Management and Control in the Cell Matrix

There are a number of examples of how the Cell Matrix can be used to manage various tasks related to its own operation and maintenance. While non–Cell Matrix systems could be designed specifically to implement any of these examples, the advantage of

the Cell Matrix architecture is that *it* supports *all* of them: the Cell Matrix architecture does not have to be modified in any way in order to implement these systems.

9.4.1 Hardware Error Checking

The wire-building and cell-testing techniques described above can be used to test individual cells within the matrix to ascertain their proper functioning. Moreover, since only basic logic circuits and simple state machines are required to perform these tests, they can be performed by circuitry within the matrix itself, which offers a number of interesting opportunities.

For example, once a small initial set of cells is known (say, via conventional validation techniques) to be functioning properly, and a state machine is built to perform subsequent cell tests, cells can be tested, verified, and then used to build longer wires, allowing testing of more-remote cells. Other than the initialization issue, this eliminates the question of "what if the test circuit itself is defective?" since only known-good cells will be used in extending the test circuitry (Durbeck and Macias 2002; Macias and Durbeck 2002, 2004).

By running multiple test circuits, cross-checking can be performed among multiple testers. Basic N-way redundancy could be used to verify the initial circuitry, after which a single copy would suffice (as far as manufacturing defects are concerned; transient errors are a different consideration).

This approach also allows the performance of parallel testing. Again, starting from a single test circuit, multiple testers can be configured from known-good cells; these testers can operate in parallel to test multiple regions simultaneously and then construct more parallel testers. This can reduce test time to $O(n^{1/2})$ for n cells in a two-dimensional matrix and to $O(n^{1/3})$ in a three-dimensional one (Durbeck and Macias 2001d; Macias and Durbeck 2002, 2004).

Test results can be stored in something similar to a "bad block" list (Duncan 1989), which can be used in subsequent configuration operations. If a place-and-route algorithm were implemented directly on the Cell Matrix hardware, it would simply note these defective cells as being unavailable for placement or routing and would thereby avoid them in creating compiled circuits.

For handling runtime defects such as single-event burnout (Waskiewicz et al. 1986) or single-event gate rupture (Fischer 1987), these tests could be performed periodically. Multiple copies of circuitry can be maintained, with copies taken offline individually, their underlying cells retested, and their configuration adjusted as needed to avoid newly defective cells.

9.4.2 Autonomous Fault Handling through Autonomous Circuit Building

We have devised a methodology for implementing a desired target circuit on top of the Cell Matrix in a way that allows that system to configure itself in order to avoid defective regions of the matrix (Macias and Durbeck 2004, 2002; Durbeck and Macias 2001a,b). If new defective regions are later found or suspected to be present (e.g., because some sort of built-in self-test has failed), the system can be given a single

REBUILD command, and it will locate, isolate, and avoid all defective regions, while reimplementing itself using only good cells. The goal is to have these operations performed by the system itself, with a minimal amount of external intervention required. This work combines the above techniques of hardware error checking with some bio-inspired concepts in self-organization (Mange et al. 2000).

This approach utilizes the concept of a Supercell, which is a general term for a collection of contiguous Cell Matrix cells configured to perform a variety of functions while still retaining the underlying self-configurability of individual cells. In our self-repairing circuit building work (Macias and Durbeck 2002, 2004) the supercell first performs a number of *initialization* functions, including:

- Testing a region of the matrix for defective Cell Matrix cells.
- Configuration of new supercells on known-good regions.
- Activation of isolation circuitry within good supercells, in order to prevent any interference from bad supercells.
- Sharing of configuration information among a network of supercells in order to configure new supercells in multiple regions in parallel.

The purpose of the initialization stage is to tile a region of the Cell Matrix with known-good supercells, while isolating defective cells. Note that this part of the system's operation requires a set of configuration strings to be sent into the empty matrix. These configuration strings depend on the high-level circuit to be implemented, but are completely independent of the location of any defects in the matrix (since the location of such defects is assumed to be unknown). All subsequent steps are performed by the collection of supercells themselves, without any further external intervention.

Following initialization, the system enters a *differentiation* phase, in which, supercells assign themselves unique integer IDs, so that they can be differentiated from each other. Without such an assignment, all supercells are identical to one another. This assignment is accomplished through the collective operation of the entire set of supercells. Differentiation changes the contents of two ID registers contained within each supercell:

- One ID contains a position-dependent integer, which is simple to assign (by incrementing an incoming neighbor's ID and passing it to other neighbors), but the set of assigned integers is not necessarily contiguous.
- A second ID that is position-independent and whose collection is guaranteed to form a set of contiguous integers.

When the initial configuration strings are developed, an abstract representation (called the "genome") of the final target circuit is coded inside the strings. The genome is simply a netlist, specifying the components of the final circuit, along with their input-to-output interconnections. What is *not* specified is the particular locations of those components in the matrix, nor the paths that will be used for their interconnections (since doing so would require knowledge of the locations of defective Cell Matrix cells).

After unique IDs have been assigned, each supercell compares its ID with the ID of components stored in the final circuit's genome (which is why contiguous IDs are

required). Using the information inside the genome, each supercell thus knows which component it is to implement in the final circuit and then configures its piece of the final circuit inside itself.

Following differentiation, the collection of supercells must be wired together in order to implement the final target circuit. This requires first determining pathways from component to component, and then creating communication channels along those pathways. Both of these steps are performed by the supercells themselves, without any external intervention. Path finding is done using a greedy algorithm, which picks the shortest path from component to component. Channel creation is performed by utilizing preexisting pieces of channels inside each supercell and by configuring cells near the junction of these channel pieces to form continuous pathways. Note that some of these channel pieces cross within the supercell to allow the creation of crossed communication pathways.

The final supercell design consisted of 270×270 Cell Matrix cells. This was intended only as a proof of concept, and represents neither the smallest nor the most efficient supercell for the given problem. There are ways to perform more traditional fault tolerance for the Cell Matrix architecture that have been developed and analyzed (Saha et al. 2004; Macias and Durbeck 2005a), but the point of the supercell work was to demonstrate autonomous, self-organizing circuitry, and faulty hardware was simply the impetus to which the system responded in order to modify its behavior.

9.4.3 Self-Replication

Figures 9.16a–e show the operation of an entirely self-replicating circuit on the Cell Matrix. The circuit is comprised of two main parts. The upper-left region of the circuit is called the main grid and is a state machine that generates bit sequences and sends them into three wires. The right half of the circuit is a copy of the left half, but with additional space (two rows of empty cells) between each row of nonempty cells. The right-hand circuit is called the exploded grid, since it is a copy of the main grid on the left, but with the rows spaced out vertically.

The reason for using an exploded grid is that we do not want the circuit to actually be operating: it is intended to supply a *data* version of the circuit being replicated. Since the circuit itself is intended to modify other cells, we want to carefully control the behavior of this copy. Thus the cells in the exploded grid that can assert their C outputs are kept permanently in C mode, so that their C outputs are constantly forced to 0, which is achieved by pairing such cells with other cells called guard cells.

The bitstreams generated by the main grid extend the three wires as follows:

- One wire is extended into the exploded grid in order to read the cells within the first row of that grid.
- Another wire is extended into an empty region below the main grid and will create a copy of the exploded grid there, but without any interrow spaces.
- The third wire is extended into an empty region below the exploded grid, and will create an exact copy of the exploded grid.

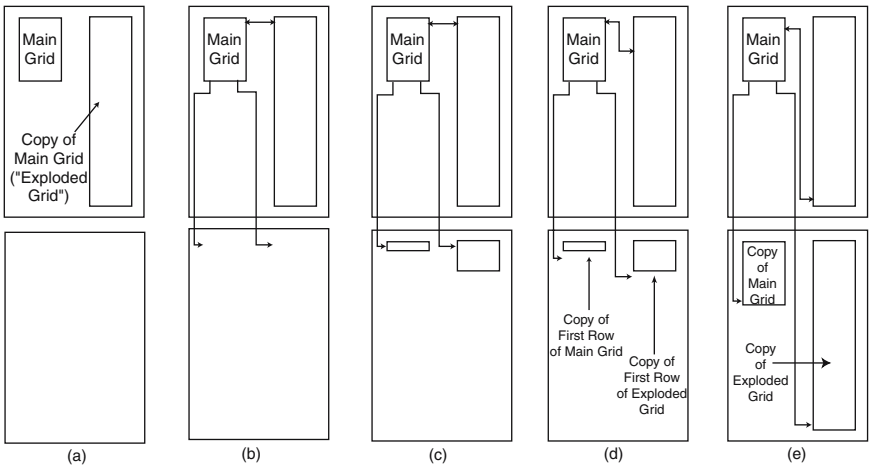


Fig. 9.16. Self-replicating circuit. (a) The initial configuration; the main grid is a circuit that will read the exploded grid and produce a copy of it and itself. (b) Three wires have been built, extending into the exploded grid and the initially empty regions to the south. (c) The first row of the exploded grid has been read and used to reproduce the first row of the main grid and the exploded grid in the region to the south. (d) The three wires are moved in preparation for reading the next row of the circuit. (e) All rows have been read and copied, creating an exact copy of the original circuit in the region to the south.

Figure 9.16a shows the initial circuit. Figure 9.16b shows the circuit with the three wires immediately prior to reading the first cell from the exploded grid. In Fig. 9.16c, the first row of the exploded grid has been completely read, and two copies of it have been made in the region below the original circuit. A single row has now been configured in the new main grid, and three rows have been configured in the new exploded grid (one row of main grid circuitry, and two rows of guard cell circuitry). This process is somewhat analogous to the translation and transcription steps found in the replication of DNA (Arms and Camp 1987). Figure 9.16d shows the circuit once the wires have been adjusted for reading the second row of the exploded grid. Of course, the wires extending into the original and new exploded grids require more extension steps than the wire in the new main grid. The above steps are repeated for each row of the exploded grid. Figure 9.16e shows the final state of the system, where an exact copy of the original circuit has been made to the south.

Typically, the lower-rightmost cell would be configured to output a GO signal into the circuit, which would trigger the execution of the above circuitry. Thus, as soon as a new copy of the circuit is created, it immediately begins making a new copy of itself. This is, in itself, not necessarily useful, but is a useful building block to which a number of various enhancements can be added. For example, after being placed on a Cell Matrix, the circuit could make a copy of itself to the south. That copy could make a copy of *itself* to the south, and that copy could make a copy of itself, and so on, thus creating a single column of copies of this circuit.

The height of the column could be hardwired into the circuit, e.g., by incrementing a counter within each circuit, and only replicating a fixed number of times. Alternatively, the height could be determined dynamically by the presence of a marker in the matrix indicating the desired extent, or the circuit can itself determine when it has reached the southern edge of the matrix (by noting that cell configuration operations no longer work). Once a leftmost column has been created, the bottom circuit can signal to the other circuits in that column to begin replicating to the east, resulting in the parallel building of a new column. Note that each circuit in the column replicates in parallel with every other circuit in that column. Thus, whereas creating the initial column (containing, say, n copies of the circuit) required n replication cycles, creating the second column requires *only one replication cycle*.

Generation of each subsequent column will also require the same time as a single replication cycle. To create an $n \times m$ array of these circuits would thus require n replication cycles to configure the first column, plus $m - 1$ replication cycles to create each of the remaining $m - 1$ columns, for a total of $n + m - 1$ replication cycles. This is extremely better-than-linear performance. In fact, configuring n copies of this circuit requires on order of only $n^{1/2}$ steps. For a three-dimensional Cell Matrix, configuration time for n copies is a mere $n^{1/3}$ steps. This is an extremely efficient way to configure large regions of a matrix.

Of course, we are usually interested in something more than simply filling the matrix with copies of a single circuit. The self-replicating circuit is intended to be a *carrier* of additional circuitry.

9.4.4 Fully Autonomous Self-Configuration

The above self-replicating circuit can be used to make copies of the supercells described previously, which will then autonomously implement a desired target circuit. Collectively, this represents a fully autonomous, fault-handling, self-configuring system. This circuit can be thought of as a seed or, perhaps, a biological cell. When a single copy of it is placed inside an empty matrix, it begins to replicate, filling a region of the matrix with copies of itself. Once a sufficient number of copies have been created, they begin to differentiate and specialize, and then work together to implement some higher-order function. Moreover, this is done in a dynamic manner, with the exact configuration of the final circuit dependent on the environment (specifically, the location of defective cells within the matrix).

9.4.5 Hardware Compilation

With the use of the techniques described above, it is possible to perform compilations of algorithms not into software, but directly into hardware. This is already an active research area of reconfigurable logic (Page 1996). However, with the Cell Matrix as the underlying hardware substrate, it is possible to design systems that are *self-compiling*, i.e., the circuitry that produces the final compiled circuit can itself be running on the Cell Matrix.

One area in which such a setup would be useful is in implementing a Just-In-Time (JIT) (Deutsch and Schiffman 1984) compilation system. There is a huge parameter space within which algorithms could be developed. For example, the word size of an arithmetic unit can be adjusted based on the characteristics of the data being processed. This might change over time, and circuit characteristics adjusted accordingly. Similarly, the number of registers available in a general-purpose processor (implemented on the Cell Matrix) or the character size of a hardware string processor could be adjusted over time. Sequences of operations that occur repeatedly could be analyzed, and new hardware synthesized to implement their collective function directly in hardware. If the circuitry is not utilized for some period of time, such hardware could be dismantled, and the underlying cells reused.

9.4.6 Hardware Operating Systems

With self-configurable hardware, it is possible to consider hardware analogues of various software concepts, especially concepts related to operating systems. This leads to the concept of a *hardware operating system*.

For example, combining wire-building techniques and bootstrap mechanisms, one can easily imagine the notion of a *hardware library*, wherein circuits consisting of Cell Matrix cells are stored, available for retrieval and replication elsewhere in the matrix, just as software libraries store code that is reused in other programs.

As another example, the notion of virtual memory could be extended to *virtual hardware*, where a matrix appears to have more hardware than actually exists. By storing cell configuration information (and utilizing appropriate compression mechanisms) and using it to configure cells as needed, it is possible to design a system on the Cell Matrix that emulates a matrix that is larger than the physical matrix on which it resides. Such a system would, effectively, intercept accesses to nonexistent cells, create them on-the-fly, and redirect requests to those newly created cells. These cells would be located virtually in a fixed location, but physically might be located somewhere else.

Closely related to this notion of virtual hardware is *hardware swapping* and *hardware timesharing*, where a single Cell Matrix is shared by multiple applications, which are loaded and unloaded from the matrix's cells, so that a single set of cells is used for more than one application. Such a system would probably employ a double-buffering mechanism, so that while one circuit is executing on one region of the matrix, another region would be configured with the second circuit to be executed. Once that circuit is ready, and the desired time slice has expired, that second circuit would begin executing, while the cells of the first circuit would be reconfigured to implement the third circuit, which would eventually be allowed to run while the fourth circuit was implemented, and so on. Once the last circuit was given a time slice, the first circuit would again be configured and allowed to run. Other than the need to double-buffer (since configuring a circuit takes a nonnegligible amount of time), this is highly analogous to swapping and timesharing of software. Again, as described above, consideration must be given to reading, preserving, and restoring not only the configuration of each cell within a running circuit, but also the state of each cell, meaning its outputs and inputs.

9.5 Status and Future Work

9.5.1 Current Status

Several specifications for the Cell Matrix architecture have been designed (Macias et al. 1999; Durbeck and Macias 2001c; Macias and Raju 2001), and these different implementations have been used in a number of software simulators (Cell Matrix Corporation 2006b). Most work to date has been done using software simulators.

Hardware implementations have also been developed, including a small custom ASIC implementation. More recent implementations have used traditional FPGAs to implement the Cell Matrix architecture (Durbeck and Macias 2001a, 2002; Macias and Durbeck 2004, 2005b), including a self-contained 8×8 Cell Matrix board called the Mod-88 (Macias and Durbeck 2005b), a 4×4 array of which has been placed on the web for use (Cell Matrix Corporation 2006c).

A complete set of tools has been constructed and placed on the web to permit anyone to construct Cell Matrix circuits and debug them using a graphical layout editor and libraries of already-built components (Cell Matrix Corporation 2006b). A place and route tool was recently developed to automatically generate layouts from circuit descriptions (Macias 2006).

9.5.2 Some Applications of Self-Configurability

There are a number of areas in which this shift from external to internal control appears especially beneficial to the design of computational and processing systems. There has been and continues to be impetus to scale systems to greater numbers of components and greater levels of complexity in the tasks they undertake. As the system's size and complexity are scaled up, the difficulties associated with managing and maintaining it also increase. When rapid increases in scaling eventually occur (Vinge 1993), it may no longer be practical to continue scaling up current strategies. Instead, new approaches to managing extreme complexity may be required.

Utilizing internal control mechanisms is a better strategy than today's external system control as computers become more complex. A key difficulty in managing extremely complex systems is the dependence on a single, centralized unit for all management tasks. In contrast, one can distribute the management and control of the system among a large number of separate management units, thereby reducing the load on each management unit, while also improving the proximity between each management unit and the circuits that it is managing. Care must be taken to avoid introducing new complexities as the number of management units is itself scaled up.

A possible solution is to use the massive resources of such large-scale systems to solve the very problem being caused by their size, i.e., tackle the problems of large-scale system design by using a large-scale system. Some areas where this may be applied are:

- Manufacturing Defects: In order to utilize many orders of magnitude more devices, computational systems such as CPUs and FPGAs will have to be able to use imperfect hardware because it will be too difficult and expensive to build perfect hardware. It thus seems that there will have to be a design shift, from systems that

are completely free of defects to systems that can handle such defects. One way to approach this challenge is to have the system itself perform initial checks on its own hardware, testing its subsystems in an efficient (parallel) manner, and noting defective regions in a way that subsequent processing can use to avoid those defects.

- Runtime Defects: Even in systems that are manufactured perfectly or whose defects have been worked around, there are still runtime defects, i.e., temporary or permanent errors that occur while the system is operating. Given the large number of components expected in future systems, the job of monitoring them for proper functioning cannot reasonably be handled from a single, centralized (i.e., external) location. Instead, the job of defect detection and mitigation may be better handled from within the system, using the large number of system components as a resource for handling the complexity of this task.
- System Design: With a jump to Avogadro-scale systems, one could expect system design times to become prohibitively long. One example of applying our thesis to this problem would be to use a massive FPGA to implement and run massively parallel CAD tools, which are then used for subsequent design work. Another is to decouple system design from system construction, so that system construction happily continues to march down Moore's law curve, while the increasingly complex task of system design has time and opportunity to develop as resources are made available.
- Initial System Configuration/Bootstrap: As systems scale to use many orders of magnitude more devices, the problem of configuring or bootstrapping them (the process of specifying their initial setup in the case of such soft hardware as FPGAs, and the bootstrapping process of invoking and initializing all processes that constitute the running system in the case of all computation systems including FPGAs and computers) rapidly becomes worse, until configuration and initialization times may be so long that systems cannot even complete initialization. Again, a possible solution is to use the massively parallel system itself as the control to implement a very powerful, parallel bootstrap system.

9.5.3 Possible Manufacturing Options

As even simple circuitry implemented on the Cell Matrix tends to consume a large number of cells, practical hardware cell matrices are difficult to create using conventional manufacturing techniques. There are, however, a number of conceivable approaches to manufacturing large cell matrices ("large" means a Cell Matrix containing a large number of cells).

Aggressive Silicon Techniques

One approach is to use aggressive techniques in silicon, such as deep deep submicron technology, while taking advantage of the fault-handling capabilities of the Cell Matrix to manage the inevitably high defect count. This, of course, requires a mitigation technique that costs less (in terms of area/cell count) than what is gained by using a very aggressive fabrication technology.

An added benefit to using cutting-edge technologies to manufacture a Cell Matrix is the possibility of using the Cell Matrix itself as a *process driver*, i.e., as a means of debugging the fabrication process itself (Durbeck and Macias 2002). As a Cell Matrix has an inherent introspection capability, it can be used to analyze the characteristics of the manufactured cells within itself, including things such as their logical behavior, the speed of their operation, the pattern of defective cells' locations, and so on. Because pathways from cell to cell are built out of cells, there are generally multiple pathways from any cell to any other cell, which means that even regions containing a large number of defects might be thoroughly analyzed (Durbeck and Macias 2002).

Wafer-Scale Integration

Another potential approach to manufacturing large cell matrices is to employ wafer-scale integration (WSI) (Saucier and Trilhe 1986; Wyatt and Raffel 1989; Fuchs and Swartzlander Jr 1992; IEEE 1995; Zeng et al. 2005), where, instead of dicing a wafer into a number of individual chips, the entire wafer is used to implement a single circuit. WSI has been explored as a means to overcome die-level defects, and in general has to address the problem of a wafer typically containing some defective dies (e.g., Saucier et al. 1988; Boubekeur et al. 1992). Various special-purpose architectures have been developed for WSI to allow operation on top of imperfect wafers (e.g., Saucier et al. 1988; Boubekeur et al. 1992), as well as a general-purpose methodology for using wafer-scale fabrication (Alam et al. 2002).

Of course, since a Cell Matrix is inherently a fault-isolating architecture, and since faults can be detected and managed efficiently using some of the techniques described above, it is an ideal architecture for using WSI.

Three-Dimensional Fabrication

While most discussion of the Cell Matrix architecture in this chapter focused on two-dimensional matrices, it is perfectly feasible to create a three-dimensional Cell Matrix, which is actually much more powerful for a number of reasons:

- Each cell is more powerful, being a six-input six-output device (assuming a cube-based structure/topology).
- Circuit routing is easier, at least for two-dimensional circuits, since, being embedded in a higher-dimensional space, nonadjacent components can be connected via the third dimension.
- For a given number of components, the maximum path length, and hence maximum delay, can be greatly decreased — e.g., a square circuit containing a trillion cells would be $1,000,000 \times 1,000,000$ cells, giving a maximum corner-to-corner path length of $2,000,000$ cells, whereas a cubic circuit containing a trillion cells would be $10,000 \times 10,000 \times 10,000$, giving a maximum corner-to-corner path length of only $30,000$ cells;
- Configuration of a three-dimensional circuit can be much faster than configuration of a two-dimensional one with the same cell count; in the above example, the two-dimensional case would require $2,000,000$ operations ($1,000,000$ to construct one

row of supercells, and another $1,000,000$ for each supercell to create a column of supercells), whereas the three-dimensional case would require only $30,000$ operations ($10,000$ to make a row of supercells; another $10,000$ to create a column of $10,000$ supercells below each of those, thus giving a two-dimensional plane of supercells; and a final $10,000$ operations for each of those $100,000,000$ supercells to create a line of supercells in the Z-axis);

- A three-dimensional matrix can be treated as a series of two-dimensional matrices with intermatrix access in the Z dimension, thus enabling techniques such as rapid context switching, parallel reading of cells, parallel writing of cells, plane-to-plane voting, and so on.
- A three-dimensional matrix has a number of two-dimensional surfaces available to the outside world, thus affording a high-bandwidth mechanism for parallel input and output of data to and from the matrix.

There have been various attempts by researchers to create three-dimensional circuitry (Seeman 1982; J. DePreitere 1994; Alexander et al. 1995; Borriello et al. 1995; Leeser et al. 1997; Meleis et al. 1997). One problem often encountered is heat buildup, but this need not be an issue with a massively parallel architecture such as the Cell Matrix, since the idea is to get algorithm speedup through the use of massively parallel algorithms, rather than through raw uniprocessor speed. Another significant impediment for three-dimensional fabrication is, again, the manufacturing defect rate, but this can be (to some degree) mitigated using Cell Matrix fault handling techniques.

9.5.4 Nanotechnology

Any discussion of high-density, three-dimensional fabrication inevitably leads to the topic of nanotechnology (Montemerlo et al. 1996; Kamins and Williams 2001; Stan et al. 2003). Roughly speaking, nanotechnology is the science of atomic-scale manufacturing. In the context of using nanotechnology to manufacture cell matrices, our primary interest is not so much in the size of the manufactured cells per se, but rather in the extremely high cell count that can be achieved because of that size. That is, our interest is not in making small cell matrices, but rather in creating cell matrices that contain a huge number of cells. This might involve manufacturing cells out of logic gates that are themselves comprised of a small number (say 10–1000) of atoms, single electrons, and soon, with each resulting cell containing fewer than a million atoms. At such a scale, we can talk about quantities such as one *mole* of cells, i.e., a number of cells equal to Avogadro's number, or roughly 10^{23} cells. Note that a three-dimensional Cell Matrix containing 10^{23} cells would have only $100,000,000$ cells along each edge.

9.5.5 Cell Matrix Support for Nanotechnology

While the Cell Matrix architecture stands to benefit a great deal from a true atomic-scale fabrication technologies, it is also possible that such technologies may also benefit from the Cell Matrix. For example, in trying to manufacture three-dimensional circuitry, one often manufactures a series of two-dimensional layers using conventional techniques (e.g., lithographic techniques), and then stacks these layers in the

third dimension. While the former process is usually quite precise, the latter is not: it involves the manipulation of macroscale objects, such as individual silicon die or silicon wafers (IEEE 1995; Ababei et al. 2004; Fraunhofer Institute for Reliability and Microintegration, Munich 2006; Misc 2006).

However, if the layers being stacked are actually cell matrices, the cells themselves can be used to create circuits on each layer that investigate their own placement relative to the layers above and below them. By staggering the placement of the cells within each layer, i.e., using nonuniform spacing between cells, it may be possible to guarantee that some of the cells will align between layers (while also guaranteeing that some cells will not). By voluntarily sacrificing some of the cells, we can ensure that some of the cells will create interlayer connections. Circuits can then be constructed to discover where these connections have been made, and that information used in the configuration of subsequent circuits.

9.5.6 Other Approaches to Manufacturing

When thinking about manufacturing a Cell Matrix, it is worthwhile to consider approaches that may lack characteristics typically required for conventional circuit manufacture. Some of the characteristics that differ for a Cell Matrix manufacturing process vs. conventional circuits include:

- Speed—a Cell Matrix is not required to have extremely fast cells, since one may hope to obtain algorithm speedup via massive parallelism rather than raw component speed.
- Power Dissipation—because, again, the individual cells can be run slowly, with overall speedup achieved via parallelism.
- Reliability—as we have seen, perfect manufacture is not strictly necessary for the creation of a viable Cell Matrix: a certain degree of defects is acceptable.
- Physical size—the primary requirement for a useful Cell Matrix is not that it be physically small, but rather that it contain a huge number of cells.

When these considerations are taken into account, there are some unconventional possibilities for manufacturing a Cell Matrix. One possibility is to use printable circuit technology (Burns et al. 2004; MacDonald 2006; Plastic Logic 2006; Sirringhaus et al. 2006; Wong et al. 2006) to create two-dimensional sheets of cells. These sheets could be folded and stacked to create narrow, but arbitrarily long matrices. Alternatively, single sheets could be stacked, and connectors pierced through the sheets to create intersheet connections. Even if only some cells are connected from sheet to sheet, this would still offer some of the benefits of a fully three-dimensional matrix.

This also leads to the notion of trying to weave a matrix, utilizing some of the ideas being used for the design of smart clothing (Cakmakci and Koyuncu 2000; Cakmakci et al. 2001; Edmison et al. 2002; Marculescu et al. 2003; Martin et al. 2003; Martin 2006). If cells can be constructed simply by controlling the pattern of a weaving, then, again, arbitrarily long two-dimensional sheets could be manufactured. This approach is worth considering if only because the production of textiles is one of the oldest manufacturing processes in human history, estimated to date back 12,000 years (Sabalan Group 2006).

Owing to of the regular structure of the Cell Matrix, it may be possible to exploit natural processes in its manufacture: e.g., the process of crystal growth. Of course, this would require a means for associating logic circuits with certain types of crystals, arranging things so that as the crystal grows, so does the matrix. A more-controlled approach is being taken in the use of DNA as a scaffolding for the placement of carbon nanotubes (Seeman 1982; Robinson and Seeman 1987; Winfree 1998; Winfree et al. 1998; Seeman 2003; Winfree 2003; Dwyer et al. 2004a,b; Kim et al. 2004; Winfree and Bekbolatov 2004; Patwardhan et al. 2004; Rothemund et al. 2004; Park et al. 2006; Patwardhan et al. 2006; Pistol et al. 2006), which could be applied to the construction of Cell Matrix cells.

9.5.7 CAD Issues—Magic Polygons

The question of how best to represent self-modifying circuitry such as that which can be implemented on the Cell Matrix remains unanswered. One idea is to use the notion of *serialization*, wherein a collection of cells is used to generate a corresponding stream of binary data. Once a circuit has been serialized, it can be processed using standard digital circuits: it can be stored, retrieved, steered through circuitry via mux/demux logic, compared with other streams, and so on. Along with serialization is the process of *deserialization*, wherein a stream corresponding to a circuit is used to configure a set of cells to implement that circuit.

Figure 9.17a shows an example of this serialization/deserialization process. A sample source circuit is shown on the right, surrounded by a dashed box called a *magic polygon,* which indicates a region of the circuit that is to be serialized. The line coming from the box transfers this stream of ordinary binary data to the deserializer on the left, where a new copy of the circuit is created in Fig. 9.17b. The (*) inside each polygon is used to position deserialized circuitry relative to the original serialized circuit. The GO signal indicates initiation of the deserialization operation.

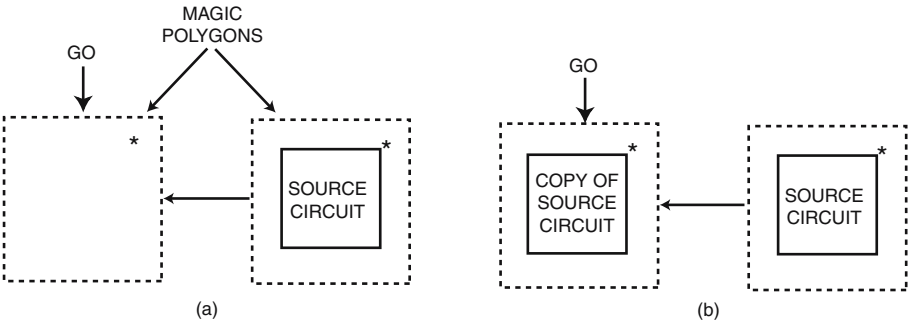


Fig. 9.17. Serializaton of a source circuit. (a) The truth tables composing the source circuit are transmitted along the wire to the deserializer on the left. The (*) is an anchor point for positioning the new circuit. (b) The deserializer has synthesized a copy of the source circuit using the serial bitstream it receives.

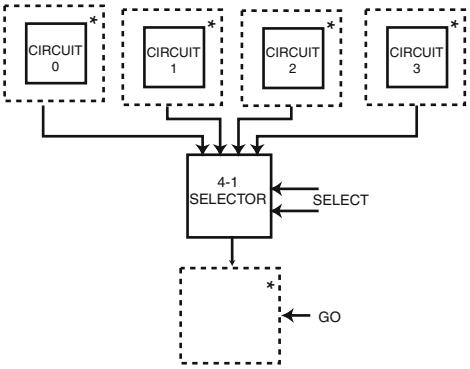


Fig. 9.18. Example of a hardware library. The 4-1 selector can choose the bitstream corresponding to one of four circuits. The selected circuit will be synthesized to the south.

Figure 9.18 shows an example of a simple hardware library, where one of four circuits can be selected for synthesis. The bitstreams for each circuit are sent into the 4-1 selector, which chooses one of them for synthesis by the magic polygon to the south.

Figure 9.19 shows a more complex example. In this case, the magic polygon on the right itself contains a magic polygon, which means that the synthesized circuit will also contain a magic polygon. When the circuit in Figure 9.19a is serialized and then deserialized, the circuit of Figure 9.19b results, which looks exactly like Figure 9.19a, except that the east-to-west wire has been extended. Each time the circuit's bitstream is deserialized, the wire is extended another step. This can be used as a mechanism for synthesizing wires, as described in Section 9.3. Note that in this example, the GO line has not been shown. The magic polygon on the left needs a GO signal in order to initiate its deserialization process, but since location its changes, the line driving the GO input also needs to be extended at each step. For simplicity, this and other details have been excluded from Figure 9.19.

Although magic polygons are an easy way to visualize the serialization/deserialization process, they are more applicable to circuit schematics than to, say, a Hardware Definition Language (HDL) description of a circuit. Ongoing research efforts seek to

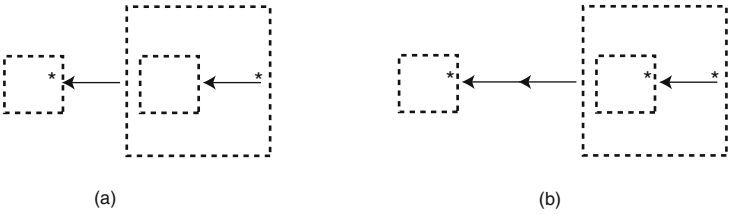


Fig. 9.19. Magic polygon representation of an extendible wire. (a) Initial setup. (b) Result after deserialization and reserialization: the wire has been extended.

elaborate on the details of magic polygons, to extend them to the realm of HDLs, and to develop tools for implementing their functional behavior.

9.5.8 A Continuous Cell Matrix

As a final point regarding possible future research, note that the Cell Matrix architecture described in this chapter is based on digital logic: while it need not necessarily be binary based, it is based on computation using some finite number (usually two) of discrete values. It is, however, theoretically possible to consider a Cell Matrix that uses *continuous* values for its inputs and outputs. For example, each input could receive a voltage between 0 and +5 V, with all values in between considered legal.

This would, however, require a continuous version of a truth table. This requires some explanation. A one-dimensional, continuous truth table (CTT) is simply a function of a single variable, representable as a two-dimensional graph. The input value appears on the X-axis, and the output value is given by the Y-axis. A two-dimensional CTT is a three-dimensional graph: the input values specify X and Y coordinates, and the corresponding Z value is the truth table's value for those inputs.

This can easily be generalized to n inputs. A real-world implementation of such a truth table is, however, a fairly difficult thing to achieve. A one-dimensional CTT might be implemented as a recirculating traveling wave, stored mechanically in, e.g., a spring or a mercury delay line in the style of older analogue memory systems. The value corresponding to an input x could be found by waiting the appropriate amount of time and then reading the output value as it recirculates from the end of the storage device back to the beginning. Implementing higher-dimensional CTTs is trickier, especially beyond three dimensions.

Of course, even in the simple one-dimensional setup, there is a nonnegligible delay associated with reading an output value from the CTT. For traditional clocked logic, this may not be a problem, but for the Cell Matrix, which inherently follows a data-flow model, it could be an issue. One solution is to make several readers available along the length of the storage device, so that rather than waiting for the desired value to reach the end of the device, it only need reach the next reader. Alternatively, the value presently at the nearest reader can be used as an approximation for the desired value, assuming certain continuity conditions on the values stored in the CTT. By changing the number of readers, a trade-off is made between the accuracy of the read value and the delay in reading it: greater accuracy can be obtained only by increasing the delay, and decreasing the delay decreases the accuracy. It is presently unclear if this is at all related to similar-sounding phenomena in the natural world (Heisenberg 1927).

Note that a continuous Cell Matrix would be extremely powerful. For example: a single cell could implement a simple amplifier by simply storing the desired transfer function in its truth table; a single-cell, configured as an analogue multiplier, would function as an AM modulator; and so on.

The question of writing and reading a CTT is also interesting. A one-dimensional CTT can easily be read and written in a fixed, finite period, by parameterizing its single input axis with a time variable: since there is only one input variable, it can simply be swept from beginning to end.

A two-dimensional CTT would require sweeping all possible combinations of *two* input variables (x and y). This suggests the use of some sort of space-filling curve, which can hit all (x, y) combinations in a finite period of time. Alternatively, the inputs can be discretized into a large (but finite) number of values, and then the space of (x, y) combinations again parameterized with time. Here again, there is a seemingly inevitable trade-off between the accuracy of the CTT lookup (due to the discretized input values) and the time required to read or write the CTT (based on the number of values in the discretized CTT).

References

ABABEI, C., Maidee, P., and Bazargan, K. (2004). Exploring potential benefits of 3D FPGA integration. *Field-Programmable Logic and its Applications*, pages 874–880. LNCS/Springer Heidelberg, Germany.

Abdi, H. (1994). A neural network primer. *Journal of Biological Systems*, 2(3):247–283.

Alam, S., Troxel, D., and Thompson, C. (2002). A comprehensive layout methodology and layout-specific circuit analyses for three-dimensional integrated circuits. *ISQED International Symposium on Quality Electronic Design, 2002*, page 246. IEEE Computer Society Washington, DC.

Alexander, M., Cohoon, J., Colflesh, J., Karro, J., and Robins, G. (1995). Three-dimensional field-programmable gate arrays. *ASIC Conference and Exhibit, 1995, Proceedings of the Eighth Annual IEEE International*, pages 253–256.

Arms, K. and Camp, P. (1987). *Biology*. Saunders, Philadelphia, 3rd edition.

Aspray, W., and Burks, A. (1987). *Papers of John von Neumann on Computing and Computer Theory*, volume 12 of Charles Babbage Institute Reprint Series for the History of Computing.

Borriello, G., Ebeling, C., Hauck, S., and Burns, S. (1995). The Triptych FPGA architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(4):491–501. IEEE Educational Activity Department Piscataway, NJ.

Boubekeur, A., Patry, J., Saucier, G., and Trilhe, J. (1992). Configuring a wafer-scale two-dimensional array of single-bit processors. *Computer*, 25(4):29–39. IEEE Computer Society Press, Los Alamitos, CA, USA.

Burns, S., Kuhn, C., Jacobs, K., MacKenzie, J., Ramsdale, C., Arias, A., Watts, J., Etchells, M., Chalmers, K., Devine, P., et al. (2004). Printing of polymer thin-film transistors for active-matrix- display applications. *Journal of the Society for Information Display*, 11:599.

Cakmakci, O., and Koyuncu, M. (2000). Integrated electronic systems in flexible and washable fibers. *None*. filed with the United States Patent Office and the European Patent Office.

Cakmakci, O., Koyuncu, M., and Eber-Koyuncu, M. (2001). Fiber computing. *Proceedings of the Workshop on Distributed and Disappearing User Interfaces in Ubiquitous Computing, CHI*.

Cell Matrix Corporation (2006a). Bibliography for Cell Matrix-related research. http://www.cellmatrix.com/entryway/products/pub/bibliography.html.

Cell Matrix Corporation (2006b). Cell Matrix Software. http://www.cellmatrix.com/entryway/products/software/software.html.

Cell Matrix Corporation (2006c). MOD 88 Online Viewer. http://cellmatrix.dyndns.org:12001/cgi-bin/mod88/obs2.cgi?

Darwin, C. (1859). *The Origin of Species by Means of Natural Selection. Or the Preservation of Favoured Races in the Struggle for Life*. Murray, London.

J. DePreitere, et al. (1994). An optoelectronic 3D field programmable gate array. In Hartenstein, W. and Servit, M., editors, *Field-Programmable Logic: Architectures, Synthesis, and Applications, Lecture Notes in Computer Science*, volume 849. Springer-Verlag, Berlin.

Deutsch, L. and Schiffman, A. (1984). Efficient implementation of the Smalltalk-80 system. *Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming languages*, pages 297–302. ACM Saltlake City, UT.

Duncan, R. (1989). Design goals and implementation of the new High Performance File System. *Microsoft Systems Journal*, 4(5):1–14.

Durbeck, L., and Macias, N. (2001a). Autonomously Self-Repairing Circuits. NASA SBIR Phase II Proposal.

Durbeck, L., and Macias, N. (2001b). Autonomously Self-Repairing Circuits. NASA SBIR Phase I Final Report.

Durbeck, L., and Macias, N. (2001c). Self-configurable parallel processing system made from self-dual code/data processing cells utilizing a non-shifting memory. US Patent 6,222,381.

Durbeck, L., and Macias, N. (2001d). The Cell Matrix: An architecture for nanocomputing. *Nanotechnology*, 12(3):217–230.

Durbeck, L., and Macias, N. (2002). Defect-tolerant, fine-grained parallel testing of a Cell Matrix. *Proceedings of SPIE ITCom*, 4867. SPIE Boston, MA.

Dwyer, C., Johri, V., Patwardhan, J., Lebeck, A., and Sorin, D. (2004a). Design tools for self-assembling nanoscale technology. *Nanotechnology*, 15(9):1240–1245.

Dwyer, C., Poulton, J., Taylor, R., and Vicci, L. (2004b). DNA self-assembled parallel computer architectures. *Nanotechnology*, 15(11):1688–1694.

Edmison, J., Jones, M., Nakad, Z., and Martin, T. (2002). Using piezoelectric materials for wearable electronic textiles. *Proceedings of the Sixth International Symposium on Wearable Computers, 2002.(ISWC 2002).* pages 41–48. LNCS/Springer Berlin, Germany.

Fischer, T. (1987). Heavy-ion-induced, gate-rupture in power MOSFETs. *IEEE Transactions on Nuclear Science*, 34(6):1786–1791.

Fraunhofer Institute for Reliability and Microintegration, Munich (2006). Department of Si Technology and Vertical System Integration. http://www.izm-m.fraunhofer.de/files/fraunhofer2/si-technology__vsi.pdf, accessed 10/31/2006.

Fuchs, W., and Swartzlander Jr, E. (1992). Wafer-scale integration: Architectures and algorithms. *Computer*, 25(4):6–8.

Haldane, J. (1931). The Philosophical Basis of Life.

Heisenberg, W. (1927). Werner Heisenberg, in a letter to Wolfgang Pauli, (February 1927).

IEEE (1989–1995). *Proceedings of the International Conference on Wafer Scale Integration*.

Kamins, T., and Williams, R. (2001). Trends in nanotechnology: Self-assembly and defect tolerance. *Proceedings of the NSF Partnership in Nanotechnology Conference*.

Kauffman, S. (1993). *The Origins of Order: Self-organization and Selection in Evolution*. Oxford University Press.

Kim, J., Hopfield, J., and Winfree, E. (2004). Neural network computation by in vitro transcriptional circuits. *Advances in Neural Information Processing Systems*, 17:681–688.

Koza, J. (1992). *Genetic Programming: On the programming of computers by means of natural selection*. Bradford.

Leeser, M., Meleis, W., Vai, M., and Zavracky, P. (1997). Rothko: A three dimensional FPGA architecture, its fabrication, and design tools. *Seventh International Workshop on Field Programmable Logic and Applications*. Springer London, UK.

Lennox, J. (2001). *Aristotle's Philosophy of Biology: Studies in the Origins of Life Science*. Cambridge University Press.

MacDonald, W. A. (2006). Advanced Flexible Polymeric Substrates. In Klauk, H., editor, *Organic Electronics: Materials, Manufacturing & Its Applications*. Wiley.

Macias, N. (1999). The PIG Paradigm: The design and use of a massively parallel fine grained self-reconfigurable infinitely scalable architecture. *Proceedings of the First NASA/DOD Workshop on Evolvable Hardware (EH'99)*. IEEE Pasadema, CM.

Macias, N. (2001). Circuits and sequences for enabling remote access to and control of non-adjacent cells in a locally self-reconfigurable processing system composed of self-dual processing cells. US Patent 6,297,667.

Macias, N. (2006). Cell Matrix place and route tool: Changes and improvements. White Paper delivered to Los Alamos National Laboratory under subcontract #90843-001-04 4x.

Macias, N., and Durbeck, L. (2002). Self-assembling circuits with autonomous fault handling. *Proceedings of the NASA/DoD Conference on, Evolvable Hardware, 2002.* pages 46–55. IEEE Washington, DC.

Macias, N., and Durbeck, L. (2004). Adaptive methods for growing electronic circuits on an imperfect synthetic matrix. *Biosystems*, 73(3):172–204.

Macias, N., and Durbeck, L. (2005a). Unpublished white papers and talks delivered to Los Alamos National Laboratory under subcontract #90843-001-04 4x.

Macias, N., and Durbeck, L. (2005b). A hardware implementation of the Cell Matrix self-configurable architecture: The Cell Matrix MOD 88. *Proceedings of the NASA/DoD Conference on, Evolvable Hardware, 2005*. pages 103–106. IEEE Washington, DC.

Macias, N., Henry III, L., and Raju, M. (1999). Self-reconfigurable parallel processor made from regularly-connected self-dual code/data processing cells. US Patent 5,886,537.

Macias, N., and Raju, M. D. (2001). Method and apparatus for automatic high-speed bypass routing in a Cell Matrix self-configurable hardware system. US Patent 6,577,159.

Mange, D., Sipper, M., Stauffer, A., and Tempesti, G. (2000). Toward self-repairing and self-replicating hardware: the Embryonicsapproach. *Proceedings of the Second NASA/DoD Workshop on, Evolvable Hardware, 2000.* pages 205–214. IEEE Paloatlo, CA.

Marculescu, D., Marculescu, R., Zamora, N., Stanley-Marbell, P., Khosla, P., Park, S., Jayaraman, S., Jung, S., Lauterbach, C., and Weber, W. (2003). Electronic textiles: A platform for pervasive computing. *Proceedings of the IEEE*, 91(12):1995–2018.

Martin, T. (2006). Tom Martin's Wearable Electronic Textiles research group at Virginia Tech. http://www.ccm.ece.vt.edu/etextiles/, http://www.ccm.ece.vt.edu/etextiles/publications/ accessed 10/31/2006.

Martin, T., Jones, M., Edmison, J., and Shenoy, R. (2003). Towards a design framework for wearable electronic textiles. *Proceedings of the Seventh IEEE International Symposium on Wearable Computers, 2003.* pages 190–199.

Meleis, W., Leeser, M., Zavracky, P., and Vai, M. (1997). Architectural design of a three dimensional FPGA. *Proceedings of the Seventeenth Conference on Advanced Research in VLSI, 1997*, pages 256–268. IEEE Computer Society HN Arbor, MI.

Misc (2006). *International Journal of Chip-Scale Electronics, Flip-Chip Technology, Optoelectronic Interconnection and Wafer-Level Packaging*. http://www.chipscalereview.com accessed 10/31/2006.

Montemerlo, M., Love, J., Opiteck, G., Goldhaber-Gordon, D., and Ellenbogen, J. (1996). Technologies and designs for electronic nanocomputers. *The MITRE Corporation, McLean, VA, MITRE Tech. Rep. MTR 96W0000044, July*.

Ortega-Sanchez, C., Mange, D., Smith, S., and Tyrrell, A. (2000). Embryonics: A bio-inspired cellular architecture with fault- tolerant properties. *Genetic Programming and Evolvable Machines*, 1(3):187–215.

Page, I. (1996). Constructing hardware-software systems from a single description. *Journal of VLSI Signal Processing*, 12(1):87–107.

Park, S., Pistol, C., Ahn, S., Reif, J., Lebeck, A., Dwyer, C., and LaBean, T. (2006). Finite-size, fully-addressable DNA tile lattices formed by hierarchical assembly procedures. *Angewandte Chemie*, 45:735–739.

Patwardhan, J., Dwyer, C., Lebeck, A., and Sorin, D. (2004). Circuit and system architecture for DNA-guided self-assembly of nanoelectronics. *Foundations of Nanoscience: Self-Assembled Architectures and Devices. Proceedings 2004*, pages 344–358. Science Technica Snowbird, UT.

Patwardhan, J., Dwyer, C., Lebeck, A., and Sorin, D. (2006). NANA: A nano-scale active network architecture. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2(1):1–30.

Pistol, C., Lebeck, A., and Dwyer, C. (2006). Design automation for DNA self-assembled nanostructures. *Proceedings of the 43rd Annual Conference on Design Automation*, pages 919–924. ACM Press New York, NY.

Plastic Logic (2006). Plastic Logic, developer of printed flexible thin film transistor (TFT) arrays. http://www.plasticlogic.com/technology.php accessed 10/31/2006.

Prodan, L., Tempesti, G., Mange, D., and Stauffer, A. (2003). Embryonics: Electronic stem cells. In Abbass, H., Standish, R., and Bedau, M., editors, *Artificial Life VIII: Proceedings of the Eighth International Conference on Artificial Life*, pages 101–105. Bradford. The MIT Press Sydney, Australia.

Robinson, B., and Seeman, N. (1987). The design of a biochip: a self-assembling molecular-scale memory device. *Protein Engineering Design and Selection*, 1:295–300.

Rothemund, P., Papadakis, N., and Winfree, E. (2004). Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12):2041–2053.

Sabalan Group (2006). Textile History. http://www.sabalangroup.com/aboutus-history-textilehist-en.html.

Saha, C., Bellis, S., Mathewson, A., and Popovici, E. (2004). Performance enhancement defect tolerance in the Cell Matrix architecture. *Proceedings of MIEL 2:* 777–780.

Saucier, G., Patry, J., and Kouka, E. (1988). Defect tolerance in a wafer scale array for image processing. *Proceedings of an International Workshop on Defect and Fault Tolerance in VLSI Systems, University of Massachusetts, Amherst, Oct.*, 8:8.2–1–8.2–13.

Saucier, G., and Trilhe, J. (1986). *Wafer scale integration*. North-Holland.

Schmit, H. (1997). Incremental reconfiguration for pipelined applications. *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 47–55. IEEE Napa, CA.

Seeman, N. (1982). Nucleic acid junctions and lattices. *Journal of Theoretical Biology*, 99(2):237–47.

Seeman, N. (2003). Biochemistry and structural DNA nanotechnology: An evolving symbiotic relationship. *Biochemistry*, 42(24):7259–7269.

Sirringhaus, H., Sele, C. W., von Werne, T., and Ramsdale, C. (2006). *Manufacturing of Organic Transistor Circuits by Solution-based Printing*. Wiley Interscience.

Stan, M., Franzon, P., Goldstein, S., Lach, J., and Ziegler, M. (2003). Molecular electronics: from devices and interconnect to circuits and architecture. *Proceedings of the IEEE*, 91(11):1940–1957.

Thompson, A. (1996). An evolved circuit, intrinsic in silicon, entwined with physics. *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware*, pages 390–405. Springer Verlag Berlin, Germany.

Trimberger, S. (1998). Scheduling designs into a time-multiplexed FPGA. *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, pages 153–160. ACM Press New York, NY.

Vinge, V. (1993). Technological singularity. *VISION-21 Symposium sponsored by NASA Lewis Research Center and the Ohio Aerospace Institute, March*.

Waskiewicz, A., Groninger, J., Strahan, V., and Long, D. (1986). Burnout of power MOS transistors with heavy ions of Californium-252. *IEEE, DNA, Sandia National Laboratories, and NASA, 1986 Annual Conference on Nuclear and Space Radiation Effects, 23rd, Providence, RI, July 21-23, 1986). IEEE Transactions on Nuclear Science (ISSN 0018-9499),*, 33(pt 1):1710–1713.

Winfree, E. (1998). Simulations of computing by self-assembly. *Caltech CS Technical Report 1998.22*.

Winfree, E. (2003). DNA Computing by self-assembly. *The Bridge*, 33(4):31–38.

Winfree, E., and Bekbolatov, R. (2004). Proofreading tile sets: Error-correction for algorithmic self-assembly. *DNA Computing*, 9:126–144.

Winfree, E., Liu, F., Wenzler, L., and Seeman, N. (1998). Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–544.

Wong, W. S., Daniel, J. H., Chabinyc, M. L., Arias, A. C., Ready, S. E., and Lujan, R. (2006). Thin-film transistor fabrication by digital lithography. In H. Klauk, editor, *Organic Materials, Manufacturing, and Applications*, Wiley VCH.

Wyatt, P. and Raffel, J. (1989). Restructurable VLSI-a demonstrated wafer-scale technology. *Proceedings of the First International Conference on Wafer Scale Integration, 1989.* pages 13–20. IEEE Computer Society Press Washington, DC.

Xilinx, Inc. (2006). Xilinx, Inc. http://www.xilinx.com accessed 10/31/2006.

Zeng, A., Lu, J., Rose, K., and Gutmann, R. (2005). First-order performance prediction of cache memory with wafer-level 3D integration. *IEEE Design & Test of Computers*, 22(6):548–555.

10

Self-organizing Nomadic Services in Grids

Tino Schlegel and Ryszard Kowalczyk

10.1 Introduction

The grid has emerged as a global platform to support on-demand computing and on-demand virtual organization for coordinated sharing of distributed data, applications and processes. The service orientation of the grid also makes it a promising platform for seamless and dynamic provision of service-oriented applications across organizations and computing platforms. Service-oriented computing enables new kinds of flexible business applications in open systems, which has changed the way of thinking about building, delivering, and consuming software over recent years. Services are made available via standard interfaces independent of their underlying platform implementations. The loose coupling, implementation neutrality, and flexible configuration of services allow the creation of large networks of collaborating applications. This computing paradigm allows companies to focus on their core competencies by providing complex business applications that are composed of their own services plus services provided by external partners. These new kinds of distributed systems are not only connected clusters of computers in a local-area network. They are loosely coupled components collaborating in a worldwide service grid within and across organizational boundaries. The traditional intraorganizational view is shifting to a global perspective.

The OASIS SOA Reference Model group defines Service Oriented Architecture (SOA) as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with, and use capabilities to produce desired effects consistent with measurable preconditions and expectations" (OASIS 2006).

The underlying fundamental principle of service-oriented computing is the exchange of standardized documents (e.g., XML documents) via standard interfaces and protocols (SOAP, HTTP, etc.) among services located on different hosts. This simple mechanism provides great flexibility with clear advantages for service grids. However, the advantages of flexibility and service distribution are often in conflict with the increased network traffic produced by highly communicative services in a distributed application. If services are located on distant servers, exchanging documents between them generates network traffic between the corresponding net-

work nodes. Every increase in network traffic causes higher execution time because of a longer time for network transmission and latency compared with a centralized solution.

Apart from the communication overhead in a service oriented environment, the organic nature of such a system demands self-managing capabilities for a seamless-link operation. Each service can be a part in a vast number of applications. Interaction partners and quality of service parameters may change over the lifetime of a service, which creates an unpredictable environment. Therefore, each individual service must be responsible for managing its own behaviour in accordance with agreements established with other services. Self-management and self-organization of services include self-configuration, self-protection, self-healing, and self-optimization which arise from more than just individual self-organizing capabilities. They constitute an emergent property of the whole system that can only be achieved by mutual interactions among services.

The communication overhead in a service-oriented computing environment requires a trade-off between the advantages of service distribution and the drawbacks regarding their execution and communication time and costs. Application data will become larger because of its human readable and interoperable XML representation and increasing multimedia content (Fontana 2004). Moreover, new user requirements introduce additional network load for meta-communication, such as semantic information about services, negotiations plus monitoring of quality-of-service parameters between services and other management and configuration information. This communication overhead has to be considered as one of the main practical downsides of the service grids despite the significant increase in network bandwidth observed over recent years (Fontana 2004).

Different communication paradigms for distributed computing have been developed over the past years to reduce the increasing network load (Bruneo et al. 2003; Braun and Rossak 2005). However, none of them can eliminate the problem, as all paradigms have advantages and disadvantages depending on a concrete application scenario. The two main communication paradigms are called *remote communication* and *mobile code*, the first one being the currently dominant one because of its simplicity, universality regarding execution platforms, and its high security. The interacting services located at different locations communicate by exchanging all information via remote messages. This can cause a huge amount of unnecessary network traffic if only small pieces of the transferred information are required. The mobile code paradigm describes the reallocation of code or parts of code to another platform for remote execution. The relocated service can communicate with a communication partner that is located on the same server locally and return only the results. Figure 10.1 illustrates the two paradigms and shows the differences between them.

It is obvious that mobile code can reduce network traffic only if the code of the service that has to be transmitted over the network is smaller than the amount of data that can be saved using this paradigm. The problem of deciding between the two communication paradigms is known as the *migration decision problem* (Braun and Rossak 2005).

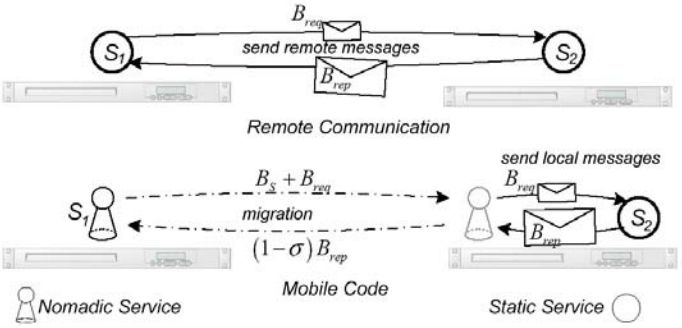


Fig. 10.1. Communication paradigms.

The software agent community developed mobile agents as an implementation of the mobile code paradigm for the optimization of network and data management in multiagent systems (Wooldridge 2002). A mobile agent has the capability to migrate with its code and execution state from host to host to fulfil its task. Mobile agents could decide to meet at an agent server and then communicate only locally without generating any network traffic except that of the migration itself (Braun and Rossak 2005). Mobile agents can not only reduce network load and increase application response time, but can also improve reliability by making the application more robust against network failures and reduce power consumption in the case of mobile devices. Many research groups have investigated and compared network communication costs of different paradigms in terms of network load and processing time in various application scenarios (Outtagarts et al. 1999; Samaras et al. 1999; Puliafito et al. 2001). The main expectation of mobile agents to reduce the network load was not satisfied and research interest in mobile agents has dwindled. Vigna (2004) said that mobile agents are very expensive, cause security problems, and in general produce worse performance than other communication paradigms. In fact, no single communication paradigm produces optimal network traffic under all conditions. Optimal performance can only be achieved with a combination of different paradigms depending on the current environment and application parameters (Strasser and Schwehm 1997).

Developments in grid computing supply excellent distributed infrastructures for the realization of service-oriented computing which are able to execute services at distributed heterogeneous resources connected by a network (Foster and Kesselman 1997; Buyya et al. 2000; Frey et al. 2002). One of the major problems in grid computing is the efficient resource allocation of tasks to available resources. Resource allocation in grids is usually done in accordance with Service Level Agreements (SLA).

Most existing grid toolkits allow the reallocation of a service during execution for runtime optimization. Services that can change their execution platform, known as *nomadic services*, can migrate at runtime to a server with more available resources for faster processing or one closer to other services to speed up network communication.

A practical example for service reallocation is the negotiation of quality-of-service parameters before the actual service execution, by exchanging offers with a number of candidates until an agreement is reached with one of them. The group of interacting services could decide to meet at the same server or cluster of servers with enough free resources for faster negotiation and only local communication. After the negotiation has finished, the services who reach an agreement can send back a message with the result only. Many distributed service grid applications could benefit from using this paradigm. However, current grid computing toolkits focus on the processing of computationally expensive jobs. They usually do not take communication costs into account and the jobs requesting resources cannot influence the resource allocation decision.

In this chapter, we propose a distributed, innovative self-organizing approach to provide intelligent communication and resource management for service grids that increases the services' self-managing capabilities. We eliminate the need for a central facilitator or resource broker. In our distributed resource allocation, it is solely the nomadic services in the system that are responsible for all resource allocation decisions. They consider the amount of available resources as well as the network transmission costs during service execution in their resource allocation decisions.

This approach is based on self-organization of nomadic services. Each nomadic service can decide between remote communication and allocation at the server of the communication partner followed by local communication, provided that enough resources are available for execution. The services learn from past allocation decisions and adapt to new environments in order to minimize network traffic, mitigate network latency, and balance resource load between the grid nodes.

Our solution uses short-term histories of the last communication acts with other services and resource load information of potential servers for an allocation of resources. Based on this purely local information, a nomadic service forecasts all environment parameters that have an impact on the communication costs of the following communication act and decides on the most beneficial communication paradigm. In addition, the resource loads of potential remote servers that provide resources are predicted. Even if a service would opt for migration when considering only communication costs, a server that is not overloaded also has to be selected. In some cases, a remote communication could be the better alternative if none of the available servers have free resources. Thus an open system needs self-healing and self-organization mechanisms for continuous optimization of the resource allocation and communication management. It takes into account service-oriented computing tendencies toward autonomy, heterogeneity, and unreliability of resources and services. The solution we present is inspired by the concept of inductive reasoning and bounded rationality introduced by Arthur (1994).

Our self-organizing approach does not require a central authority for controlling, decision support, or information sharing among services. The resource allocation in the system is created by the effective competition of services for available resources and is a purely emergent effect. We demonstrate in various simulation experiments that the services can self-organize in a dynamic service grid environment without the need for active monitoring or a central controlling authority.

10.2 Related Work

The grid community has developed grid toolkits such as Globus (Foster and Kesselman 1997) and Condor-G (Frey et al. 2002), which provide middleware infrastructures for service oriented computing. These toolkits focus mainly on effective resource allocation of computationally very expensive jobs without addressing communication costs. In contrast, service grids are transaction intensive with a large number of services that are not computational expensive.

Existing distributed approaches for resource management in multiagent systems can avoid server overloading by refusing or queuing incoming mobile agents (Fluess 2005). From the system perspective, the problem of load balancing among different agent servers is important, and existing solutions to this problem provide a good supplement to our self-organizing resource allocation.

Most of today's techniques for resource scheduling found in grid computing toolkits like Globus or Condor-G use a coordinator instance such as an auctioneer, arbitrator, dispatcher, scheduler, or manager that needs to have global knowledge concerning the state of all resources. These resource allocation mechanisms are appropriate for building self-contained software systems, but they are not designed to face the challenges of open and dynamic environments, where resources can be added or removed at any time. The Cactus project (Allen et al. 2001) focuses on dynamic resource allocation in grids using a central resource directory to discover and assign available resources, which causes performance problems in large-scale grids.

Recent research in grid computing has recognized the value of decentralized resource allocation mechanisms and has investigated a number of approaches based on economic market models for trading resources and services for the regulation of supply and demand inspired by principles of real stock markets (Clearwater 1996; Buyya et al. 2002). These approaches use different pricing strategies such as a commodity market model, posted price models, or different auction methods. Users try to purchase resources that are required to run the job cheaply, while providers try to make as much profit as possible and operate at full capacity. Buyya et al. (2002) presented a market based economy framework for resource allocation based on the regulation of supply and demand (Buyya 2002) for the grid toolkit Nimrod-G (Buyya et al. 2000), having a main focus on job deadlines and budget constraints. Even if the decision-making process is distributed, these kinds of approaches use a central facilitator during the resource allocation process (Buyya et al. 2000). Another mainly unsolved problem of these approaches is the fine-tuning of price, time, and budget constraints to enable efficient resource allocation in large systems (Wolski et al. 2001).

Some approaches for resource allocation in grids, such as the Agent based Resource Allocation Model (ARAM) are designed to schedule jobs using agent technology. Agents located on each resource cooperate in order to allocate jobs for efficient execution. The main drawback of this model is the extensive use of messages for periodic monitoring and information exchange within the hierarchical structure. Subtasks of a job migrate through the network until they find an available resource that meets the price constraints. This migration itinerary is determined by the resource connection topology (Manvi et al. 2005).

Considerable work on decentralized resource allocation techniques using game theory has been published in recent years. Most of the techniques are formulated as repetitive games in an idealistic and simplified environment (Arthur 1994; Challet and Zhang 1997; Grosu et al. 2002; Galstyan et al. 2003). A self-organizing resource allocation approach for sensor networks based on reinforcement learning techniques with a focus on optimizing energy consumption for the network nodes is presented in Mainland et al. (2005).

The problem of network communication cost optimization has been addressed in isolation by other research groups. Empirical evaluations have shown which communication paradigm performed better regarding network load and processing time in various application scenarios (Outtagarts et al. 1999; Samaras et al. 1999; Puliafito et al. 2001). Depending on the sizes of the exchanged documents, the code sizes, and the network environment one can decide whether the mobile code paradigm or the remote communication paradigm performs better. Most current solutions to the migration decision problem are based on mathematical models of the application's network load using parameter estimations (e.g., number of communication steps and average document sizes) and suggest a decision at design time. We believe that this type of solution is not sufficient in an open and distributed application scenario. It will be beneficial to address the migration decision problem at runtime, because only then will all influencing parameters (network throughput, latency, document and code sizes) actually be known or can best be approximated.

To illustrate this problem, we use the simple model presented in Braun and Rossak (2005) (compare Fig. 10.1). In the case of remote communication, a request message of size B_{req} is sent to a remote service, which answers with a result message of size B_{res}. The total network load is $B_{RC} = B_{req} + B_{res}$. In the case of migration, nomadic service S_1, which contains code of size B_C and state information of size $B_S + B_{req}$ migrates to the remote server followed by local communication with service S_2. Service S_1 can process the result and extract only necessary information or compress the reply message by a factor of σ ($0 \leq \sigma < 1$), so that only $(1-\sigma) \cdot B_{res}$ must be carried back to the home platform. The service does not carry its code or the request message back because the code is already available at the home server and the request message is obsolete. However, new state information must be carried back, which accumulates this network load to $B_{MC} = B_C + 2 \cdot B_S + B_{req} + (1-\sigma) \cdot B_{res}$. An evaluation of the model using an artificial parameter setting shows that there is a breakeven point B^*, at which the migration overhead produced by the code and state relocation is exactly compensated for by the amount of data reduction (compare Fig. 10.2).

Picco (1998) advocated estimating all influencing application parameters at design time then deciding between the two paradigms. This approach is suitable for simple scenarios in which all the parameters are known to have constant size. Also in case that the size of the parameters (e.g., the result size) is randomly Poisson distributed with parameter λ, and $B^* \ll \lambda$ or $B^* \gg \lambda$, which indicates that the probability for a wrong decision is very low, a static decision is sufficient. In more complicated and dynamic situations a static decision on communication paradigms is prone to error.

Strasser and Schwehm (1997) have improved this static analysis by providing an algorithm to determine the optimal itinerary for a mobile agent for n servers by

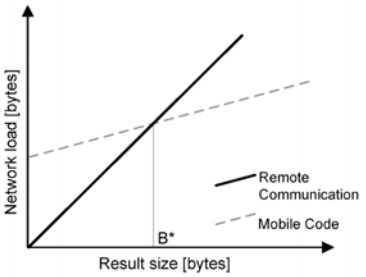


Fig. 10.2. Evaluation of the model for both communication paradigms.

allowing a combination of both paradigms. In their approach, the agent only migrates to a subset of all servers, whereas others are contacted using remote communication under the assumption of full knowledge. This approach was further improved by collecting all necessary information about network quality using distance maps before planning the optimal migration itinerary (Theilmann and Rothermel 2000), which increased the network load of the system significantly due to active monitoring data.

10.3 Communication Cost Optimization and Resource Allocation

The basic mechanism of our solution is inspired by techniques from adaptive and complex systems using inductive reasoning and bounded rationality principles. A complex adaptive system is "a dynamic network of many agents acting in parallel, constantly acting and reacting to what the other agents are doing. The control of a system tends to be highly dispersed and decentralized. If there is to be any coherent behaviour in the system, it has to arise from competition and cooperation among the agents themselves. The overall behaviour of the system is the result of a huge number of decisions made every moment by many individual agents" (Waldrop 1992).

10.3.1 The El Farol Bar Problem

Arthur (1994) introduced an ill-defined decision problem, called the El Farol bar problem, which assumes and models inductive reasoning. It is probably one of the most studied examples of complex adaptive systems. In the bar problem, 100 people decide independently each week whether to go to a bar on a certain night. Space is limited and people can enjoy the evening only if the bar is not too crowded, defined as fewer than 60 people in attendance. There is no way to be sure of the number of people in advance. All people have the same preferences and they so if they think that they will enjoy themselves. No communication is allowed among patrons and choices are unaffected by their individual experiences. The only information available to all the people is the attendance figure for the past weeks. This problem has no perfect, logical, and rational solution. If all believe that few people will go, all will go, and if all believe that

most people will go, nobody attends the bar, invalidating the belief in both situations. The only way it can work is if people's expectations differ.

The solution is derived from the human way of deciding ill-defined problems. Humans tend to keep in mind many hypotheses and act on the most plausible one. Therefore, each agent keeps track of the performance of a private collection of its belief models (predictors) and selects the one that is currently most promising for decision making. At the beginning of the simulation, each person is assigned a number of predictors randomly from some predefined set. One of these, called the active predictor, is used to predict the next week's attendance, which is the basis for the decision to go or stay at home. After all the agents have made the allocation decision, the accuracy of every predictor is calculated. The predictor with the best accuracy for the last n weeks becomes the new active predictor for each person. For initialization, reasonable accuracies and past attendance information are chosen. The result of Arthur's simulation is that the attendance figures show a fast stabilizing number of people attending at around the optimal number of 60. Challet et al. (2004) have analyzed this problem and presented a rigorous mathematical model.

Distributed resource allocation in an open and dynamic environment is a similar problem that cannot assume or guarantee perfect rationality. Services cannot rely on other services they are dealing with and they are forced to guess at their behavior. Methods that are needed in such a scenario are not deductive, but inductive.

10.3.2 Adaptation to a Service-Oriented Computing Scenario

Assumptions and constraints of the service grid scenario cannot be directly mapped to the model presented by Arthur (1994). Our assumptions differ as follows:

1. We do not allow free information dissemination about past server utilization. Nomadic services learn from their own past experience, which means that each service only has information about server utilization for the times when it was located at the server. The service has no information about resource utilization at other times; in particular the service cannot validate allocation decisions not to go because it has no information for assessment.
2. We know neither the number of available system resources and its capacities nor the number of services competing for resources. This means that the system must adapt to situations with different numbers of available resources and nomadic services.
3. We do not assume that all services make synchornized decisions at the same time.
4. We do not have any information on which to base the initialization of the history at the beginning of the life cycles.

10.3.3 Detection of Service Providers in a Service-Oriented Environment

Services have to know the locations of other services that are needed to accomplish their goals. We assume that each nomadic service has a list with locations of services that can be used. The problem of service discovery is a separate one and not within

the scope of our research. However, there are different ways in which services can be retrieved, such as a central service directory (yellow page service) or more sophisticated distributed solutions dependent on the environment and requirements. Services can exchange such location information about other services with their communication partners or browse the service directory when they migrate to another server.

10.3.4 Model

We model a distributed service grid environment as a set of servers $L = \{l_1, \ldots, l_n\}$ and nomadic services $S = \{s_1, \ldots, s_n\}$, each located on its home server $hs(s_i)$ at the beginning of its life cycle. The map $\mathfrak{L} : (S, t) \rightarrow L$ defines the location of each service at time t in general. Services bound to large databases or otherwise immobile legacy systems are called stationary and denoted S^s. Each service has to process a list of communication steps $CS =< c_i \mid i : 1..p >$ as part of its task. A communication step $c_i :=< s_a, s_b, m_k, m_j >_i$ defines that a request message m_k is sent from service s_a (source) to service s_b (destination), which responds with a reply message m_j. Each message m can be seen as an arbitrary sequence of bytes of length $B_m(m_k)$. The network cost for remote communication comprised of a request message and a reply message is calculated using Eq. (10.1). The cost for the mobile code paradigm with the migration of two nomadic services s_a and s_b to a remote server is calculated by Eq. (10.2). Both services have to carry the codes of size B_C. The requesting service additionally has to carry the request message m_i, and the results of size $(1 - \sigma) \cdot B_m(m_j)$ on the way back. If a service migrates to the server of its communication partner, the code size of the service not migrating has to be disregarded:

$$B_{RC}(c_i) = \begin{cases} 0 & if\ \mathfrak{L}(s_a) = \mathfrak{L}(s_b) \\ B_m(m_k) + B_m(m_j) & \text{otherwise} \end{cases} \tag{10.1}$$

$$B_{MC}(c_i) = \begin{cases} 0 & if \mathfrak{L}(s_a) = \mathfrak{L}(s_b) \\ B_C(s_a) + B_m(m_k) + B_C(s_b) + (1 - \sigma) B_m(m_j) & \text{otherwise} \end{cases}. \tag{10.2}$$

Each server l_i has a capacity $\mathfrak{C}(l_i, t)$, which may vary over time. Each nomadic service consumes $U(s_i, t)$ server resources for its execution, independent of the executing server. The resource load $\mathfrak{U}$ of server l_i at time t is calculated using Eq. (10.3):

$$\mathfrak{U}(l_i, t) = \sum_{m=1}^{n} U(s_m, t) \mid \mathfrak{L}(s_m, t) = l_i. \tag{10.3}$$

10.3.5 Algorithm Description

This section describes our self-organizing, distributed resource allocation algorithm for nomadic services, which also optimizes the network communication costs of the system. The algorithm is implemented in each nomadic service and is based on beliefs

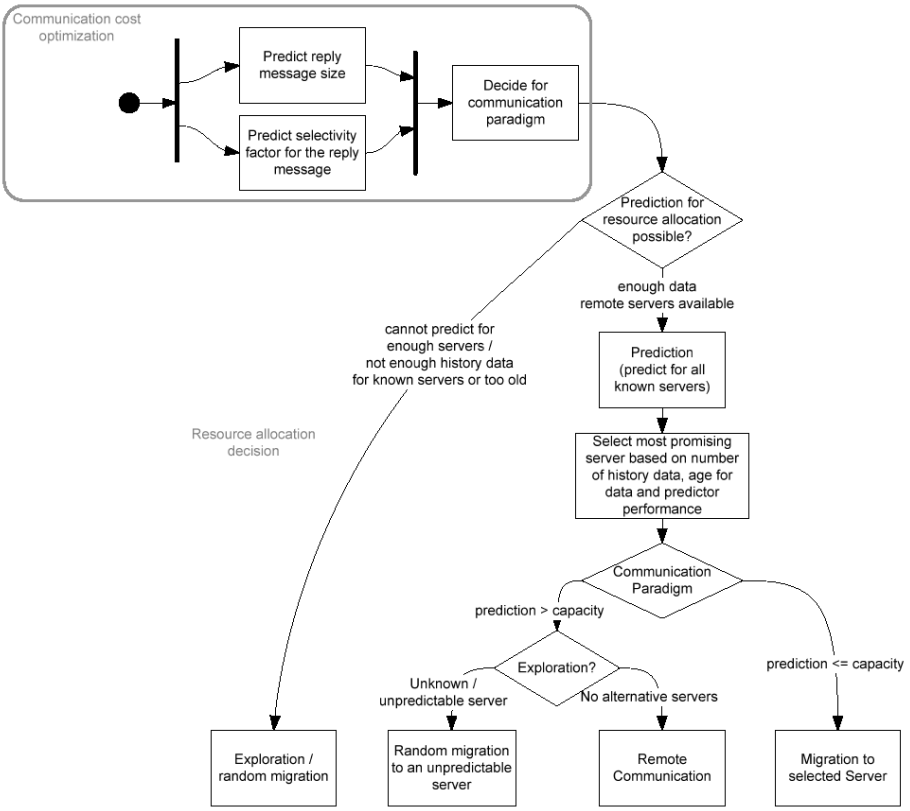


Fig. 10.3. Graphical visualization of the distributed, self-organizing algorithm of a service.

about their environment. The beliefs are modelled using forecasts of environment parameters based on individually observed data. A visualization of the decision making of the algorithm is illustrated in Fig. 10.3. The forecast of the necessary environmental parameters requires short-term histories H of these parameters [Eq. (10.4)]. Each history item $h_i = (x_i, y_i)$ is a pair comprising the date x and the value y. The most recent history value is h_0:

$$H_m(l) = (h_0, ..., h_i) \mid 0 \leq i < m. \tag{10.4}$$

All nomadic services use a set of different predictors for forecasting each parameter. Each predictor in such a predictor set is a function $p : H \rightarrow \mathrm{N}^+ \cup \{0\}$ from the history space to a positive integer that is the predicted value. An example predictor for the prediction of the resource utilization is, for instance, the average resource utilization that the nomadic server observed over the last five executions at this server.

All predictors of a set $\mathcal{P} := \{p_i \mid p \in P \wedge i = 1...k\}$ are randomly chosen from some predefined set of predictors. Each set has one active predictor $p^A \in \mathcal{P}$ that makes the next step's prediction. After each decision, all predictors in the set are

evaluated based on their predicted values and a new active predictor is chosen. The nomadic service's decision-making process and the selection mechanism of the new active predictors are described in more detail in the following section.

10.3.6 Communication Costs

The optimization of network communication costs considers only predicted costs for the next communication act. Before the next communication act c_{i+1}, a nomadic service decides which communication paradigm produces the lower network load. This decision requires predictions for two unknown parameters—the reply message size and the message selectivity (Fig. 10.3: communication cost optimization). The active predictors in the set for these parameters will predict the sizes based on the current historical information. Based on the predictions, the nomadic service calculates the expected network load for both paradigms using Eqs. (10.1) and (10.2) and selects the better communication paradigm. After the communication act, all predictors of both sets are evaluated based on the accuracies of the predictions, new active predictors for both sets are chosen, and history information of both parameters is updated.

The predictor with the smallest accumulated absolute error R over the last l predictions of each set becomes the new active predictor. The absolute error of a prediction is calculated using Eq. (10.5).

$$R = \sum_{i=0}^{l-1} \delta_i; \quad \delta(p^C) = \ |y - \tilde{y}\ |, \tag{10.5}$$

where y is an actual value, and $\tilde{y}$ is the predicted value.

In our first experiments we were keen to learn about the quality of very simple predictors. Actually, it is not as important to achieve a precise prediction of the environment parameters as to make the correct decision based on the predicted values.

All experiments for predictions of environment parameters for the communication cost optimization use the following types of predictors:

- Same value as n^{th} last communication act: $p^C(n) = y_n$.
- Mean value of last n communication acts:

$$p^A(n) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} y_i.$$

- Linear regression over the last n communication acts: $p^L(n,t) = a \cdot t + b$, where a and b are calculated using linear regression with least-squares fitting of the last n history values against their observation date.
- Gaussian distributed random value of last n communication acts:
  $p^G(n) = N(\mu, \sigma^2); \qquad \mu = \frac{1}{n} \cdot \sum_{i=0}^{n} y_i; \sigma^2 = \frac{1}{n} \cdot \sum_{i=0}^{n} (y_i - \mu)^2$

The technique of using a set of predictors provides good results as the best predictor will be chosen automatically while the less accurate predictors are not considered

in the decision process. Specialized predictors can easily be implemented to recognize application-specific patterns to increase the accuracy and improve the decision-making process. To reduce the computational overhead required for evaluation of all predictors, a user-defined threshold d can be nominated that evaluates the active predictor's performance only until the relative prediction error ε exceeds d.

10.3.7 Self-organizing Resource Allocation

Before a nomadic service considers a migration to a remote server, it evaluates whether or not the mobile code paradigm is beneficial in terms of network communication costs. Only if the mobile code paradigm is beneficial or the network communication costs are not considered, is migration and allocation of resource at other servers an option. In this section, we focus on the resource allocation algorithm and assume that a nomadic service is willing to allocate resources at a remote server.

The basic prediction mechanism for the distributed resource allocation is similar to the communication cost optimization mechanism. The main difference lies in the selection of the active predictor, which is nondeterministic in the algorithm for resource allocation. This is to prevent the invalidation of the nomadic services beliefs, which can occur because the allocation decisions of nomadic services competing for a resource are not independent of each other. Competing nomadic services must have different beliefs of the future resource utilization to prevent the previously described invalidation of beliefs and allow a successful adaptation of their strategies to changing environments. A probability distribution over all predictors in a set is created based on the predictor efficiency values $E(p^R)$ over the last l predictions. The selection of the new active predictor is implemented as a roulette-wheel selection according to this distribution (Fig. 10.4). The predictor efficiency is a measure for the correctness of the decisions that a nomadic service made based on the predictions. The probability of selecting a predictor as a new active predictor is zero if the predictor cannot predict a value based on the current historical information. This happens in the case in which not enough historical information is available. Equation (10.6) is used to calculate the predictor efficiency of a resource utilization predictor, which is the sum of the predictor efficiency ratings of the last l predictions. A positive efficiency rating is given if the prediction led to a correct decision, a negative rating for each prediction that led to a wrong decision, and a neutral rating in the case that the decision cannot be assessed

Fig. 10.4. Example probability distribution of a predictor set.

due to a lack of information. The prediction accuracy is not considered for the resource allocation as this will not improve the resource allocation decisions. Imagine a server with a capacity of 100 resource units of which 90 are occupied. Suppose a nomadic service with a resource requirement of two units for its execution predicts 103 occupied resource units. Such a prediction which indicates overutilization and that an allocation of resources at this server would be dismissed. This is worse than a prediction of 15 occupied resource units, which is less accurate but leads to a correct allocation decision. Therefore, the efficiency of predictors is used for the selection of the active predictor:

$$E(p^R) = \sum_{i=0}^{l-1} e_i, \tag{10.6}$$

where

$$e_i = \begin{cases} 1 & \text{if} \quad i\text{th decision was correct} \\ 0 & \text{if} \quad i\text{th decision had unknown outcome} \\ -1 & \text{if} \quad i\text{th decision was wrong} \end{cases}.$$

Figure 10.4 shows an example of a probability distribution of a set of resource utilization predictors, which was created using efficiency ratings. Although predictor $P9$ has the highest efficiency in the set, predictor $P5$ was chosen as the active predictor. This nondeterministic selection mechanism enables the selection of different active predictors, which leads to different resource utilization predictions for nomadic services even though the set of predictors is the same.

For faster adaptation of the system to a changing environment (different resource capacities, number of servers, or nomadic services) and to reduce unnecessary data overhead, only the last l predictions are considered for the calculation of efficiency.

Each nomadic service uses 10 simple predictors per set. All predictors are chosen randomly from a pool of 34 predictors in total. These predictors are chosen from a predefined set which includes all predictor types that are used for the communication cost prediction (ref Sec. 10.3.6) with different cycles or window sizes as well as the following two types pf predictors:

- n-frequency distribution predictor: $p^F(n)$ uses a random value from the frequency distribution of the n last history values.
- n-mirror predictor: $p^F(n) = 2 \cdot \overline{H} - y_i$ uses the mirror image around the mean value of all history values of the nth last history value.

There is no resource utilization information about other servers available at the beginning of the simulation. Therefore, the algorithm has an initialization phase in which the resource utilization information about other servers is collected. A nomadic service migrates randomly to one server and allocates resources. If a nomadic service never migrates to the server, the resource utilization prediction mechanism and especially the self-organization would never start working due to a lack of historical information. If resource utilization predictions are available, a nomadic service will select a server with expected free resources or choose remote communication with the partner from its current location. The resource utilization prediction can only be evaluated if the

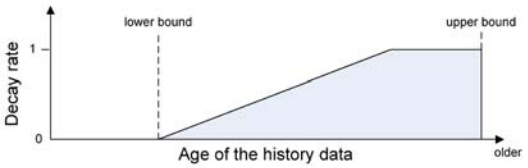


Fig. 10.5. Evaporation of old historical information.

nomadic service migrates to the servers and has the actual resource utilization information. In the case of remote communication, a majority voting of all predictors is used for the evaluation of the all predictors. In addition, old historical information of all servers is deleted using a decay rate dependent on the age of the data. The decay rate increases linearly with the age of the historical data. Recent history information has zero probability below the lower bound and probability equal to one for information older than the upper bound (Fig. 10.5). The decay of old historical information is necessary to allow an adaptation to a dynamic environment. A nomadic service that predicts a server to be overloaded may predict overload in the future because it uses the same historical information, and this can only be prevented by updating the historical information. Predictions are only as good as the historical information that is provided. Very old historical information may not allow a reflection of the current system state and is removed.

In most cases, a nomadic service can choose from a set of servers to which it could migrate. In a service-oriented environment one service is usually provided by a number of different service providers. A nomadic service can choose among those providers based on their available resources. The service predicts the resource utilization of all potential servers and selects one server from the set of servers that are predicted to have free resources. The selection depends on the predictor set efficiency calculated using Eq. (10.7). It considers the amount of historical information about the server, the average age of the historical information, and the efficiency of the active predictor. These values are transformed into a probability distribution and one server is selected using a roulette wheel selection algorithm:

$$C(\mathrm{P}) = w_1 \cdot \frac{size(H)}{m} + w_2 \cdot \frac{\overline{Age(H)}}{\max\left(\overline{Age(H)}\right)} + w_3 \cdot \frac{C(p)}{\max\left(C(p)\right)}, \qquad (10.7)$$

where w_i are the weights; $w_i \geq 0 \ \wedge \ \Sigma w_i = 1$; $size(H)$ is the number of data in history; m is the maximal number of history values; and $\overline{Age(H)}$ is the average age of historical data.

10.4 Performance Evaluation

The performance of the proposed algorithm was evaluated in various simulation experiments. A number of results presented in the next section show the behaviour of the dis-

tributed self-organizing resource allocation in different simulated environmental conditions with and without the optimization of communication costs of the system. All experiments were conducted in a special test bed developed in the Java programming language, independent of concrete grid toolkits. The simulation environment enabled different configurations for all model parameters that influence communication costs and resource allocation, such as, e.g., the number of servers, resource capacities of the servers, the number of static and nomadic services, and the number of communications steps. A discrete event simulation model was used to trigger all events that change the state of the system. The conducted experiments were divided into sections that focus on different aspects of our model and its behaviour in different environmental situations.

The first set of experiments shows the performance of the communication cost optimization algorithm. It focused on the prediction accuracy of the unknown parameters that were necessary to select the communication paradigm that produces less network traffic. Network protocol overhead for TCP or IP was not considered.

The other experiments show the behaviour of the resource allocation algorithm in isolation as well as in combination with the communication cost optimization. We demonstrate the effective self-organization of nomadic services when they compete for a single server with constant and limited resources. Experiments in a multiserver environment with different constant resource capacities follow. Experiments in a dynamic environment report on the adaptability of the model to changing resource capacities and varying numbers of nomadic services.

The presented results show the behaviour for one representative experiment and mean values and standard derivation from 100 repeated experiments for the average case. The resource utilization of each server was calculated using Eq. (10.3). Even if resources were limited, we did not limit the number of services that could migrate to a server in our simulation experiments. Therefore, in some cases the resource utilization was higher than the actual server capacity. This can be easily avoided by queuing incoming services or specifying the server capacity slightly below the actual capacity to avoid this situation.

The following attributes were used to assess the performance of our approach:

- Server resource load: The development of the resource utilization of all servers was measured in resource units over the simulation time.
- Communication paradigm selection (system view): The absolute number of services using remote communication or migration to a remote server over the simulation time.
- Communication paradigm selection (service view): The accumulated number of migrations and remote communications per agent over the whole experiment.

The following parameters influenced the simulation experiments model and were used for the performance assessment:

- Number of nomadic services: Experiments have been conducted with different numbers of static and nomadic services between 100 and 2000.
- Resource consumption: Nomadic services consumed resources during execution at a server. To measure server resource utilization, we assigned a value for the

resource consumption during execution to every nomadic service. This value corresponds to real world metrics such as memory or processor cycles. All experiments used variable values for the resource consumption that were assigned from a specified interval before each execution.

- Server: Servers were resource providers and accommodated services. Nomadic services could migrate there and allocate resources for a limited amount of time. All servers host a static service, which was the communication partner of all nomadic services. The resource consumptions of these static services were not considered as they did not influence the resource allocation itself. The home servers of nomadic services were also not incorporated into the resource allocation process.
- Execution time: The time that was needed for the execution of the nomadic service, independent of the execution platform.
- Time between executions: The amount of time that lies between two executions of a nomadic service. The default value was zero, which means that a nomadic service restarts itself immediately after it has finished execution. The default value was used in most experiments and had a major influence on the age and amount of the historical data about servers.

10.5 Simulation Results

10.5.1 Communication Cost

This section reports on results of the communication costs optimization algorithm. The prediction technique for the two unknown environment parameters performed well, especially in a dynamic environment as we used a set of predictors instead of only one. In fact, it could keep pace with every static analytical approach. Figure 10.6 shows the development of the reply message size that varies over time and the corresponding predicted values. The reply message was generated by a Gaussian distribution with increasing mean μ over the first 150 time units, beginning with 15 kbytes and later dropping. The variance σ^2 of the Gaussian distribution was kept constant at 2450 bytes over the whole experiment. We were keen to learn about the quality of simple

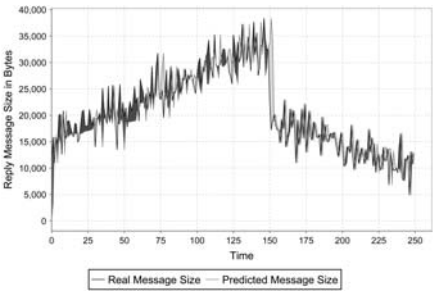


Fig. 10.6. Prediction of a variable reply message size.

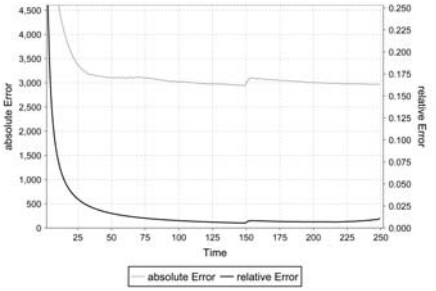


Fig. 10.7. Average absolute and relative error of the reply message size prediction.

predictors. All services had a set of 15 randomly assigned predictors. We observed that the most predictions were made by mean and $(1, 2)$-cycle predictors as they provided the most accurate results for this distribution. Figure 10.7 shows average relative and absolute errors of the predictions. It was expected that the error was close to the standard derivation of the distribution, which can be seen in Fig. 10.7. Application scenarios with communication costs that are not close to the breakeven point of both paradigms lead to correct decisions using such simple predictors. However, the quality of the predictors influences the prediction and some application scenarios require specialized predictors which are easy to integrate in to the corresponding nomadic services. The prediction mechanism leads to an effective optimization of the network communication costs due to the automatic selection of the beneficial communication paradigm.

10.5.2 Resource Allocation for a Single Server

The following experiments focus on the decentralized, self-organizing resource allocation algorithm. All nomadic services opt for a migration to the remote server and allocate its resources if they expect free resources to be available. Communication costs were not considered in these experiments. The first experiment shows results in a single-server environment with a limited constant resource capacity of $C(l_1) = 1000$ resource units. The number of nomadic services was 100 with one static service s_1^s located on server l_1. The home servers of the nomadic services were not considered. Figure 10.8 illustrates the experimental setup of the simulation environment. The resource consumptions $U(s_i, t)$ of all nomadic services were randomly assigned from the interval $[5, 50]$ resource units. Execution time of all nomadic services was 1 time unit. Figure 10.9 illustrates the development of the server utilization for one representative experiment. The provided capacity of the server was utilized to a satisfying level. Not many resources were wasted and only a few situations of overestimation occurred. The available resources in this experiment were very limited. As indicated in Fig. 10.11, only approximately 30 out of 100 nomadic services could be executed simultaneously on the server. The services could self-organize well even if resources were very limited. Figure 10.13 shows the selection of the server from the nomadic services' view. The self-organization was fair as there was no group of services that

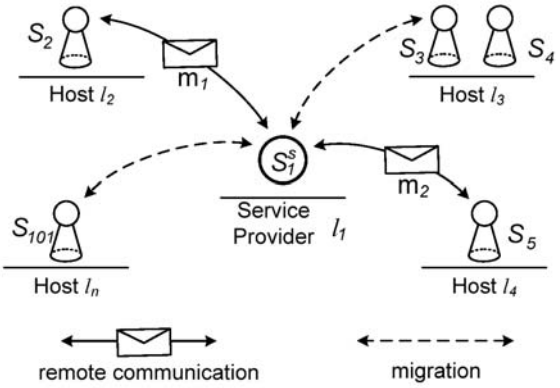


Fig. 10.8. Communication model with multiple nomadic agents and a single remote server with limited capacity.

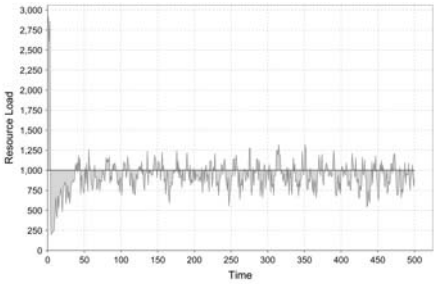


Fig. 10.9. Resource load development of a single server over 500 time units from one representative experiment.

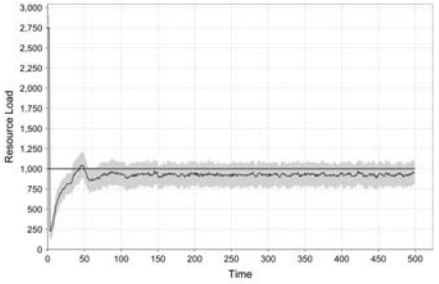


Fig. 10.10. Average resource load development (mean and standard derivation) over 500 time units.

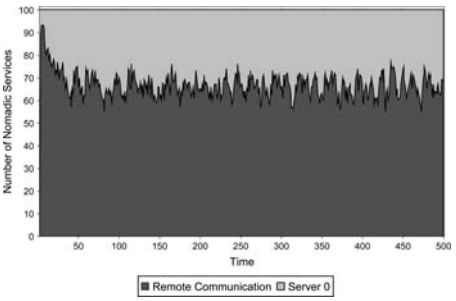


Fig. 10.11. Development of migration vs. communication of the nomadic services over time in one experiment.

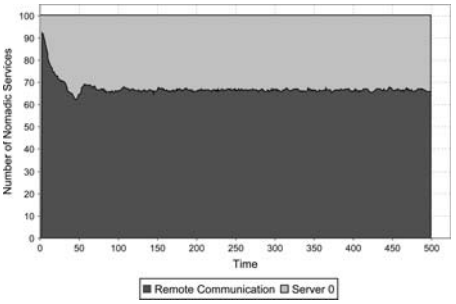


Fig. 10.12. Average development of migration vs. communicating nomadic services over time and 100 experiments.

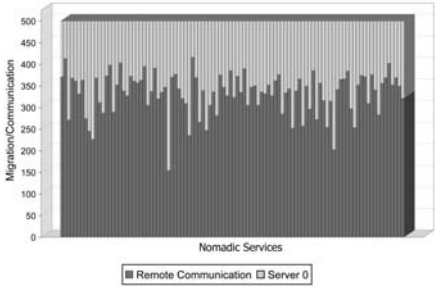


Fig. 10.13. Accumulated number of migration and remote communication per agent over 500 time units.

always migrated to the server while others had no chance of using remote communication because there were no free resources. The utilization chart in Fig. 10.10 over 100 repetitions of this experiment shows that the average server utilization was close to the optimal value of 100 % with only small variance. Figure 10.12 shows the average numbers of migrations vs. remote communications. Figure 10.14 illustrates the active

predictor usage statistics, which show how often a predictor was selected as the active predictor. It is obvious that the predictors based on mean values were favoured, while others types of predictors were not considered (mirror predictors) for decision making.

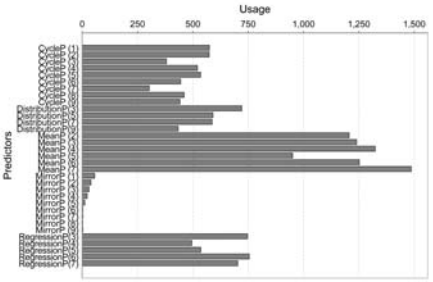


Fig. 10.14. Active predictor usage statistics accumulated over all nomadic services for 500 time units.

We also conducted experiments with different server capacities ranging from 500 to 5000 resource units. All the experiments showed that this simple case with one server can be handled very well if the provided resource capacity allowed the simultaneous execution of 20 % or more of nomadic services at the server in such a stable environment. Below this threshold, not enough up-to-date historical information was available, which led to frequent random migrations and overutilization of the server resources. Decreasing the decay rate of historical data improved this situation in the stable environment, but led to problems in more dynamic scenarios.

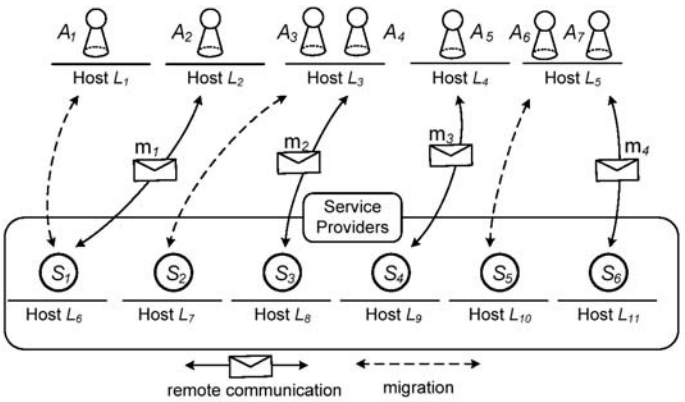


Fig. 10.15. Communication model in a multiple-server environment.

10.5.3 Distributed Resource Allocation for Multiple Servers

For the experiments in a multiple-server environment we used a similar setup, but with the difference that the static service s_1^s was provided by six different service providers. All nomadic services knew those service providers. The only difference among all the providers was a different resource capacity. Figure 10.15 illustrates the experimental setup used for the following experiments, in which communication costs were not considered.

The first set of experiments reports on results in a stable multiple-server scenario with constant resource capacities $\mathcal{C}(l_i)$ of 3500, 1350, 2500, 2500, 1500, and 10,000 resource units, respectively, and a number of 800 nomadic services. The resource consumptions $U(s_i, t)$ for all nomadic services were randomly assigned from the interval of $[5, 45]$ resource units. The execution time was assigned from the interval $[1, 10]$ time units. After completing the execution, all nomadic services were restarted immediately. The total capacity, of all servers was slightly higher than the average amount of simultaneously requested resources by all nomadic services. The development of the average resource utilization of all six servers is shown in Fig. 10.16. After around 100 time units all nomadic services self-organized their resource allocation requests in this environment and the resource utilization of each server was stable with only slight variations. It can be observed that servers with low capacity operated at full capacity while other servers had free resources available. This happened because the objective for nomadic services was a resource allocation at a server that was not overloaded, which was achieved, rather than to balance the load of all the servers equally. Figure 10.17 shows the number of services using the mobile code paradigm vs. remote communication. Almost all nomadic services migrated to one of the servers, as enough resources were provided by all service providers. On average, nomadic services made 160 resource allocation decisions over the duration of 600 time units during the experiment. One interesting result of the self-organization was the migration frequency to each server. It can be observed that nomadic services migrated to each server according to capacity, as shown its in Fig. 10.18. Some might expect that services had "favorite" servers that were used most as they were best predictable based on the most up-to-date historical information. However, nomadic services migrated to all servers according to the resource capacity distribution of the latter, which is represented in the layered structure shown in Fig. 10.18.

10.5.4 Distributed Resource Allocation in a Dynamic Server Environment

This experiment demonstrates the adaptability of our resource allocation algorithm to varying server capacities in a multiple-server environment. The servers had different initial resource capacities of 2500, 1500, 2500, 2500, 3000, and 10,000 resource units, respectively. After an initialization period of 150 time units, the capacities of four servers began to change over time. Server 2 (Fig. 10.19(b)) and server 6 (Fig. 10.19(f)) changed their capacities periodically over time. The capacities of server 3 and server 5 remained constant. Server 1 decreased its available amount of resource to 2000 resource units (Fig. 10.19(a)), server 4 (Fig. 10.19(d)) increased the initial resource

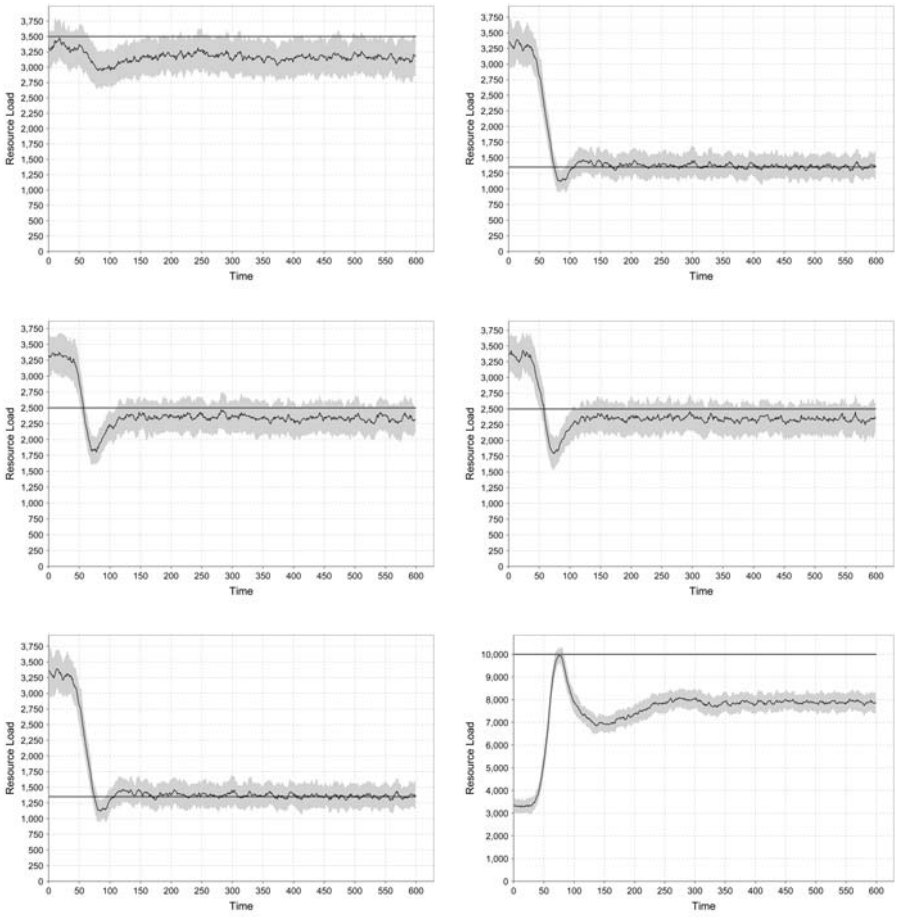


Fig. 10.16. Average resource load development of six servers over 600 time units (mean values and standard derivation).

capacity to 4000 resource units. A number of 2000 nomadic services competed for the available resources in this experiment. Values for the nomadic services' resource consumptions were unchanged in the interval $[5, 45]$ units. The execution time of each service was in the interval of $[1, 10]$ time units. Nomadic services had a break between two executions randomly assigned from the interval of $[1, 15]$ time units.

It can be seen that this dynamic environment was more challenging for our distributed resource allocation algorithm. The resource allocation mechanism required a constant adaptation to new environmental conditions. Figure 10.19 also shows that the initial adaptation period for all servers increased compared to the previous experiment as this amount of time was required for collecting historical information about other resources. After about 200 time units, almost all nomadic services adapted to the

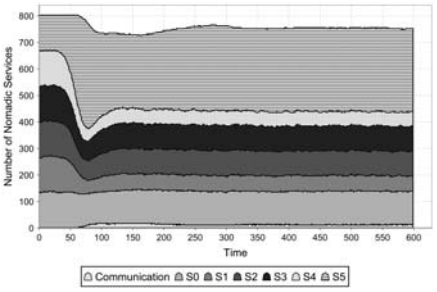


Fig. 10.17. Communication paradigm selection (system view).

multiple-server environment and utilized the available server resources optimally. Figure 10.21 shows the number of services that migrated to one of the servers or used remote communication. On average, only some 900 of the 2000 nomadic services were simultaneously active, which can be seen in Fig. 10.20, which also shows that nomadic services migrated to other servers when their amount of available resources increased. However, some nomadic services used remote communication even when free resources were available. The resource load development in Fig. 10.19 shows that the adaptation to different capacities was good and very fast. The adaptation was excellent especially in cases of a periodic and smooth alteration. We ran experiments with sudden changes in the server capacity, which showed worse adaptation as this sudden change can be compared to the initialization period in a new environment. The most difficult case for nomadic services is the exploration of additional resources of servers already visited. This is because they only learn about additional resources when they visit the server the next time. Therefore, each nomadic server should decide between exploration of new resources by random migration to such servers and an exploitation of reliable servers with free resources.

10.5.5 Resource Allocation with Communication Cost Optimization

The last experiment reports on results of our decentralized self-organizing resource allocation algorithm in combination with the optimization of the communication costs.

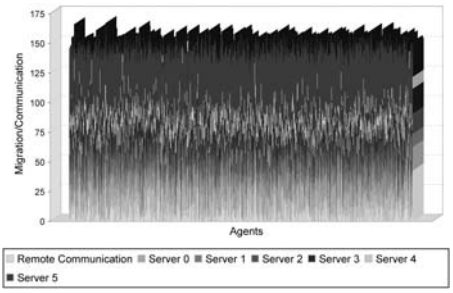


Fig. 10.18. Communication paradigm selection (service view).

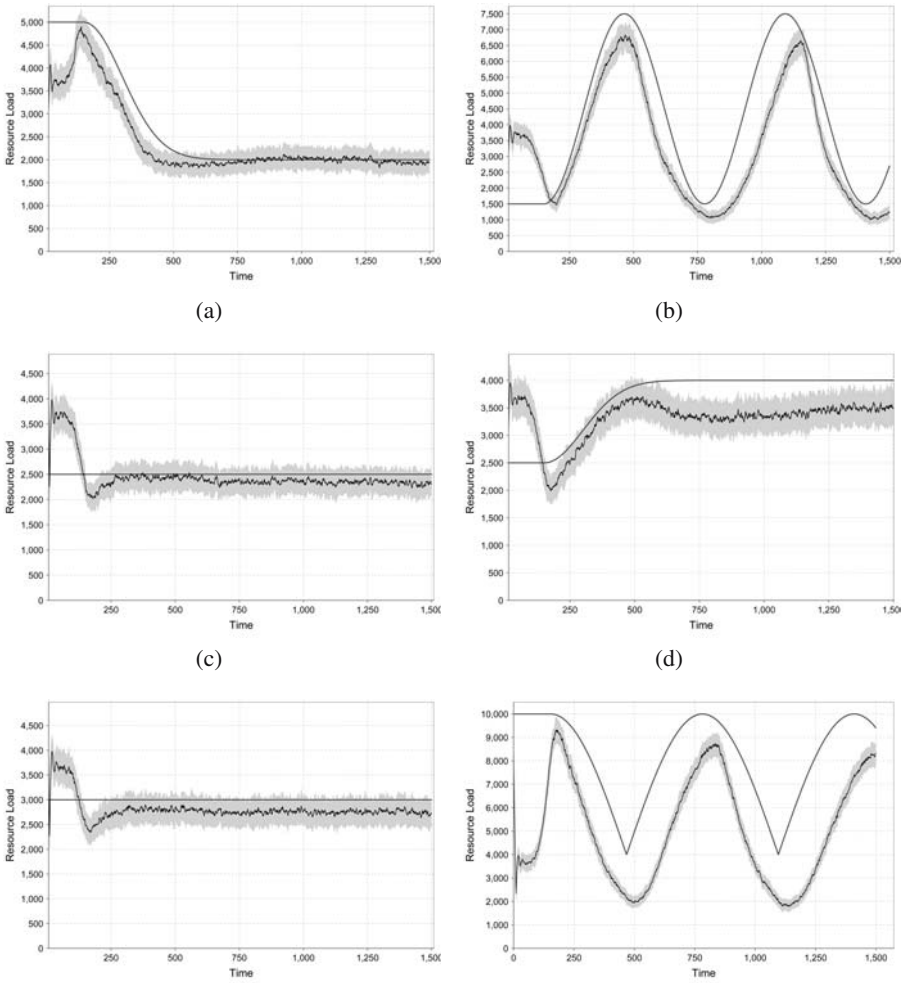


Fig. 10.19. Resource load development in a dynamic environment.

The experiment was conducted with two servers of constant capacity of 1000 units and 300 nomadic services. The initialization period was different from prior resource allocation experiments as all services used remote communication as the default communication strategy until the communication costs were predictable. The execution time and time between two executions were randomly assigned from the interval $[1, 15]$ units, and the resource consumption was between $[5, 45]$ units. The communication cost parameters were randomly initialized as Gaussian-distributed random values with different values for μ and σ^2.

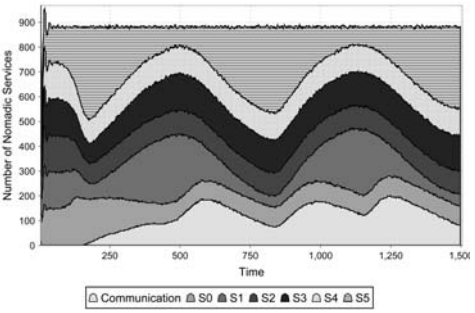


Fig. 10.20. Communication paradigm selection (system perspective).

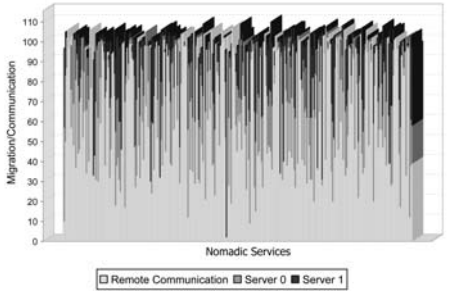


Fig. 10.21. Communication paradigm selection (service perspective).

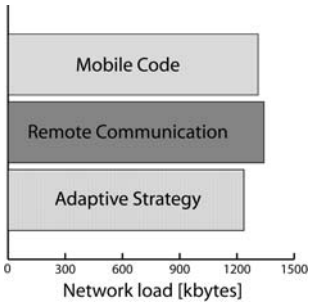


Fig. 10.22. Comparison between communication paradigms.

Neither server was fully occupied as the number of services that prefer mobile code was below the provided capacities. Figure 10.21 shows the average number of remote communications vs. migrations per service. It can be observed that some nomadic services used remote communication in most cases, as this paradigm was the best trade-off between the network load and available server resources. The number of migrations was equally distributed between the two servers.

Figure 10.22 shows the comparison between selecting the communication paradigm for all nomadic services and our adaptive strategy. Choosing either remote communication or mobile code was not the best solution, as expected. Even if the difference was small, applying our adaptive strategy could reduce the network load. A scenario with a bigger difference between the two paradigms would be more beneficial in terms of the network traffic reduction with our proposed adaptive strategy.

10.6 Conclusions

This chapter described a distributed, self-organizing resource allocation mechanism for a service grid environment in combination with network communication cost optimization. Nomadic services act purely on local information that is learnted from previous communication acts and resource allocations at remote servers. In order to optimize the communication costs, the communication paradigm that is expected to produce less network traffic is used for the next communication act. If the mobile code paradigm is beneficial in terms of the network load, the nomadic services balance the available resource capacities by choosing the server that has the most likely free resources available. The prediction mechanism stores short-term histories of environment parameters and forecast values for the decision making in the next step. This mechanism was inspired by inductive reasoning and bounded rationality principles. All control is distributed among nomadic services in the system. No additional control mechanism or management layer is required. The resource allocation is a purely emergent effect created by the effective competition of nomadic services for the system is resources. We demonstrated in various simulation experiments that the proposed system can adapt to changes in the environment by adjusting the services' behaviour at runtime.

In our solution, nomadic services adapt the use of different preassigned strategies, but they never change or update them. In biological self-organized systems, interactions among individuals or the behaviour of individuals may change slowly over generations. In our system the services' behaviours always stay the same. An update of the nomadic services' behaviours over their lifetime might be beneficial and needs further investigation.

The proposed solution has no central or hierarchical structure and therefore no classical scalability problems if more services and servers join the system. We discovered problems of a different nature. The initialization period increases linearly with the number of potential servers for the allocation of resources in the case that the total amount of provided resources is lower than the average amount of requested resources. Imagine an environment with a large number of potential resource providers and all nomadic services can choose among all of them. In the initialization period, services will randomly migrate to any of the available servers. Owing to a lack of resource capacity, nomadic services cannot find any server that is not crowded and explore all other servers to gather more information, but, they cannot find any if all nomadic services continue to migrate randomly. Before all services find out that there are no free available resources, historical information is already outdated. In fact, there is no way to gather the resource utilization information of many servers. Additionally, the decay

rate for historical information must be decreased manually in our current implementation to prevent the too frequently random migration to the potential servers. Another straightforward solution is the limitation of the number of potential servers for resource allocation.

In our current implementation the decay rate for old historical information is a static predefined function. A dynamic environment requires up-to-date information, whereas a stable environment can be predicted with old historical data. Depending on the environmental conditions and execution frequency of a nomadic service, one should alter the decay rate to improve the resource allocation performance.

We tried different types and different numbers of predictors per nomadic service and learned that there is no optimal set of predictors for resource allocation. It is important that services have a selection of different types of predictors that can predict a range of values above and below the resource capacity to prevent invalidation of their beliefs.

This kind of distributed resource allocation based on self-organization requires many nomadic services with numerous and repeated interactions. Interactions in our algorithm are indirect when nomadic services compete for a resource at the same time. Science grids that usually run few computational very expensive tasks are not suitable for this kind of self-organizing resource allocation algorithm.

References

Allen, G., Angulo, D., Foster, I., Lanfermann, G., Lui, C., Radke, T., Seidel, E., and Shalf, J. (2001). The Cactus worm: Experiments with dynamic resource selection and allocation in a grid environment. *International Journal of High-Performance Computing Applications*, 15(4), 345–358.

Arthur, W. B. (1994). Inductive reasoning and bounded rationality. *American Economic Review (Papers and Proceedings)*, 84(2), 406–411.

Braun, P., and Rossak, W. R. (2005). *Mobile Agents–Basic Concept, Mobility Models, and the Tracy Toolkit*. Morgan Kaufmann San Francisco, CA, USA.

Bruneo, D., Scarpa, M., Zaia, A., and Puliafito, A. (2003). Communication paradigms for mobile grid users. In the *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2003 (CCGrid 2003), 669, Tokyo, Japan.

Buyya, R. (2002). *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. Monash University, Melbourne.

Buyya, R., Abramson, D., Giddy, J., and Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing. Special issue on grid computing environments, *Journal of Concurrency and Computation: Practice and Experience*, 13–15(14), 1507–1542.

Buyya, R., Chapin, S., and DiNucci, D. (2000). Architectural models for resource management in the grid. In *Proceedings of the First International Workshop on Grid Computing*, 18–35, Springer, LNCS, Bangalore, India.

Challet, D., Marsili, M., and Ottino, G. (2004). Shedding light on El Farol. *Physica A*, 332, 469–482.

Challet, D., and Zhang, Y. C. (1997). Emergence of cooperation and organization in an evolutionary game. *Physica A*, 407(246).

Clearwater, S. H. (1996). *Market-based Control. A Paradigm for Distributed Resource Allocation*. World Scientific, Singapore.

Fluess, C. (2005). *Capacity Planning of Mobile Agent Systems Designing Efficient Intranet Applications*. PhD Thesis, Universität Duisburg-Essen, Germany.

Fontana, J. (2004). S*ervice-oriented Hype to Meet Hard Realities*. Network World. http://www.networkworld.com/news/2004/11010

Foster, I., and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputing Applications*, 11(2), 115–129, MIT press; ISSN 1078–3482.

Frey, J., Tannenbaum, T., Foster, I., Livny, M., and Tuecke, S. (2002). Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5(3), 237–246.

Galstyan, A., Kolar, S., and Lerman, K. (2003). Resource allocation games with changing resource capacities. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, 2003, Melbourne, Australia, July 14-18, 2003*, 145–152, ACM Press, New York.

Grosu, D., Chronopoulos, A. T., and Leung, M.-Y. (2002). Load balancing in distributed systems: An approach using cooperative games. In *Proceedings of the International Parallel and Distributed Processing Symposium*, Ft. Lauderdale, FL. IEEE Computer Society, Washington, DC, USA, 196.

Mainland, G., Parkes, D. C., and Welsh, M. (2005). Decentralized Adaptive Resource Allocation for Sensor Networks. In Proceedings of the 2nd USENIX Symposium on Network Systems Design and Implementation(NSDI '05), May 2005, Boston, MA, USA.

Manvi, S. S., Birje, M. N., and Prasad, B. (2005). An agent-based resource allocation model for computational grids. *Multiagent and Grid Systems—An International Journal*, 1(1), 17–27.

OASIS. (2006). Software Oriented Architecture (SOA) Reference Model. Available at http://www.oasis-open.org/committees/download.php/19361/soa-rm-cs.pdf, 15.09.2006.

Outtagarts, A., Kadoch, M., and Soulhi, S. (1999). Client-server and mobile agent: performances comparative study in the management of MIBs. In *Mobile Agents for Telecommunication Applications, Proceedings of the First International Workshop (MATA 1999), Ottawa (Canada), October 1999*, World Scientific Publishing.

Picco, G. P. (1998). *Understanding, Evaluating, Formalizing, and Exploiting Code Mobility*. Politecnico di Torino.

Puliafito, A., Riccobene, S., and Scarpa, M. (2001). Which paradigm should I use? An analytical comparison of the client-server, remote evaluation and mobile agent paradigms. *Concurrency and Computation: Practice and Experience*, 13(1), 71–94.

Samaras, G., Dikaiakos, M. D., Spyrou, C., and Liverdos, A. (1999). Mobile agent platforms for web-databases: A qualitative and quantitative assessment. In *Proceedings of the First International Symposium on Agent Systems and Applications* (ASA'99)/*Third International Symposium on Mobile Agents* (MA'99), Palm Springs, CA, IEEE Computer Society Press, Washington, DC, USA.

Strasser, M., and Schwehm, M. (1997). A performance model for mobile agent systems. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA'97), pages 1132–1140, CSREA Press, Las Vegas, NV, USA, June 30–July 3, 1997.

Theilmann, W., and Rothermel, K. (2000). Dynamic distance maps of the internet. In *Proceedings of the 2000 IEEE INFOCOM Conference*, Tel Aviv March 2000 IEEE Computer Society Press, Washington, DC, USA, pages 275–284.

Vigna, G. (2004). Mobile agents: Ten reasons for failure. In *Proceedings of the 2004 IEEE International Conference on Mobile Data Management* (MDM'04), pages 298–299. IEEE Computer Society, Los Alamitos, CA, USA.

Waldrop, M. M. (1992). *Complexity: The Emerging Science at the Edge of Order and Chaos*. (1st Edition) Simon and Schuster, New York.

Wolski, R., Plank, J. S., Brevik, J., and Bryan, T. (2001). Analyzing market-based resource allocation strategies for the computational grid. *International Journal of High Performance Computing Applications*, 15, pages 258–281, Sage Science Press.

Wooldridge, M. (2002). *An Introduction to Multi-Agent Systems*. Wiley. John Wiely & Sons, Chichester, England.

11

Immune System Support for Scheduling

Young Choon Lee and Albert Y. Zomaya

11.1 Introduction

Haven't there been enough approaches to scheduling problems? In terms of variety the answer might be 'yes.' However, in terms of effectiveness the answer is not as straightforward. In many scheduling problems, it is highly improbable, if not impossible, to obtain optimal schedules within a reasonable amount of time in spite of the adoption of a wide range of approaches, including evolutionary computation (EC), artificial neural networks (ANN), fuzzy systems (FS), simulated annealing (SA), and Tabu search (TS).

In recent years attention has been drawn to another biologically inspired computing paradigm called artificial immune systems (AISs). An AIS abstracts and models the principles and processes of the biological immune system in order to effectively tackle challenging problems in dynamic environments. Major AIS models include negative selection, clonal selection, immune networks, and more recently danger theory (Garrett 2005). There are some similarities between these principles and processes in the immune system and those found in other nature-inspired computing approaches, EC and ANN in particular. However, there are also substantial differences. In particular, adaptive cloning and mutation processes make AIS distinctive and useful.

Two fundamental immune activities are the recognition and the elimination of foreign agents irrespective of their previous exposure to the host. One can easily notice the inherent applicability of these robust and adaptive immune functionalities to many hard problems, such as pattern recognition, anomaly and fault detection, and machine learning. Although not as directly applicable, scheduling can also benefit from AIS, as illustrated in recent scheduling strategies (King et al. 2001; Costa et al. 2002; Coello Coello et al. 2003; Engin and Doyen 2004; Swiecicka et al. 2006).

Due to the NP-hard nature of scheduling problems, in most cases heuristic approaches have been studied extensively. In general, these heuristics are susceptible to dynamic environments, which motivates the search for more robust alternatives, and some effective strategies have already been proposed in the literature. Since environmental flux is ubiquitous in natural environments, many of the proposed approaches are inspired by nature; hence the name nature-inspired computing. AIS as a new breed

of this soft-computing paradigm has been showing potential in scheduling as well as many other application areas.

It is noted that the AIS community is actively striving to broaden the application areas of AIS. In addition, there is an increasing amount of research that attempts to better exploit the rich set of immune components, principles, and processes. Some outcomes from theses attempts (KrishnaKumar et al. 1995; Dasgupta et al. 1999; Hajela and Yoo 1999; Feng and Feng 2004) have been integrated with other approaches, such as genetic algorithms (GA) and FS to further improve their performance.

11.2 Scheduling Problem

Scheduling is the process of allocating a set of resources to tasks or jobs in order to achieve certain performance objectives and satisfying certain constraints. These scheduling objectives include minimizing schedule length (SL), also called *makespan*, minimizing response time, and maximizing resource utilization. Temporal and resource constraints are two primary conditions imposed on scheduling. For example, there may be a specific execution order that the tasks must follow and a resource that can be used for no more than one task at a time.

Due to its importance the scheduling problem has been studied extensively in the context of many disciplines, such as operations research, manufacturing, computer science, and economics. There are a number of different scheduling problems including multiprocessor, job shop, and flow shop scheduling. Although these various scheduling problems are taken into consideration throughout this chapter, the main focus is on the multiprocessor scheduling problem. Hereafter, scheduling denotes multiprocessor scheduling unless stated otherwise.

Broadly, scheduling problems are classified as static or dynamic (Grama et al. 2003). They are distinguished by the time at which the scheduling decisions are made. With static scheduling, the necessary information, such as the processing requirements of tasks and the processing capacities of resources are identified and schedules are determined a priori. Conversely, scheduling information in a dynamic scheduling scheme is obtained on-the-fly. Dynamic scheduling attempts to reduce scheduling overheads as well as job completion time. Both of these scheduling models have been studied widely and intensively. The two most common scheduling strategies are heuristics and suboptimal approximation techniques, such as randomized search methods.

11.2.1 Multiprocessor Scheduling

Ideally, it would be expected that the execution time of a job, consisting of a set of tasks, in a computer system with m processors would be m times faster than in a single-processor computer system. However, this is not quite true in practice, because generating an optimal schedule of the partitioned tasks is NP-hard; and the partitioned tasks of a job may not be completely independent (Grama et al. 2003). There may be additional challenges, such as resource and task heterogeneity; and the uncertainty of resource availability and capability, which further complicate scheduling.

Scheduling approaches include list scheduling, approximation, and random guided search. As the scheduling problem is NP-complete, algorithms that generate near optimal schedules have a high time complexity. Conversely, for any upper limit on time complexity, the quality of the schedule in general will also be limited. Together, these suggest a trade-off between performance and time complexity over the class of all scheduling problems.

11.2.2 Scheduling Heuristics

Heuristics are popular because in most cases they deliver good solutions in less than polynomial time. A primary intention of heuristics is to find a solution as fast as possible, potentially at the cost of quality. Heuristics are characterized by their essentially deterministic operation: the choice of solutions to a scheduling problem is not stochastic.

List scheduling heuristics represent the single dominant heuristic model. This is because empirically, list-scheduling algorithms tend to produce competitive solutions with lower time complexity compared to algorithms in the other categories (Kwok and Ahmad 1998). The two phases commonly found in list scheduling are task prioritization and processor selection. The tasks in the task graph are first assigned priorities using some prioritization method and are kept in a list in decreasing order of priorities. In the processor selection phase, each task is assigned to an appropriate processor based on the processor selection policy of the scheduling algorithm. List scheduling heuristics can be further improved by incorporating other techniques, such as task insertion and task duplication. With the task insertion method a task is allocated to the processor's earliest start time slot as long as it does not violate the precedence constraints. Here, the allocated slot might be between two previously assigned tasks. The rationale behind duplication-based scheduling algorithms is to increase processor utilization and decrease the communication overhead by duplicating tasks on different processors.

An alternative scheduling heuristic is clustering, where each task is initially regarded as a cluster. If the merging of two clusters produces a shorter finish time they are merged. This process repeats until no further merging can occur. After the clustering is done the tasks in each cluster are assigned to the same processor in order to reduce the communication overhead.

11.2.3 Randomized Search Techniques

Many available randomized search algorithms do not generate completely random schedules—they utilize information from previous search paths—but they do make decisions stochastically which shape the search paths. Randomized search techniques can usually be interpreted as a biased random walk over the state space of all possible schedules. Typical examples are GA, SA, and TS. Despite their high time complexity they are robust and adaptive because they do not make assumptions about the structure of the problem.

GAs are inspired by the process of biological evolution, where natural selection operates on a population of phenotypes, resulting in differential reproduction that transfers the essential structures of the most successful genotypes to the subsequent generation. The main steps involved in a GA are: (1) An initial population is randomly generated. (1) The entire population is evaluated by a fitness function and the best solutions are selected based on their fitness values. (3) The selected solutions are then mutated and/or recombined, which gives a new generation of individuals. (4) Steps 2 and 3 are repeated until a termination condition (e.g., a fixed number of generations has elapsed with no improvements on the best solution) has been reached. The mutation operator is the source of the randomized search decisions. The performance of a GA is sensitive to the value of its control parameters, such as population size, crossover, frequency, and mutation frequency.

Like GAs, SA repeats a series of processes until the termination criteria is satisfied. At each iteration step, the algorithm randomly chooses a neighbour of the current state and always moves to the neighbour if it is an improvement on the value of the current state. If it is worse than the current state, the algorithm may still move to the new state with some probability. Initially this probability is high, allowing free movement across the state space, but over time the probability diminishes according to a "cooling" schedule. The performance of SA is significantly affected by the neighbour selection method and the cooling schedule. Unlike GAs, SA has been shown to converge to the optimal solution given infinite time.

TS searches neighbours of the current solution like SA at each step. As it prevents cycles in search paths, it may produce an approximation to the optimal solution more efficiently. TS works by forbidding moves to states that have been visited within a fixed number of previous steps.

11.3 The Immune System

The immune system is a biological defence mechanism designed to protect an organism primarily from microbes, such as bacteria, archaea, fungi, protists, and viruses. An allied force of cells, tissues, and organs battles these foreign invaders. Although at first glance the immune system easily performs its core functions of detecting and killing infections, that is made possible only by the careful coordination of various immune entities incorporated with immune principles and processes. Some examples of these immune functionalities are pattern recognition, memory, learning, negative selection, and clonal selection. At the highest level, two lines of defence (the innate and the adaptive immune systems) are embodied. The core forces of both systems are different types of white blood cells.

11.3.1 Innate Immune System

The innate or nonspecific immune system is the first line of defence that uniformly combats invaders directly and immediately with chemical substances and specific types of white blood cells. As its name implies, this innate immunity already exists

at the time of birth and is triggered to respond against known invading entities. Typical examples of the former chemical substances are skin, saliva, tears, sweat, and gastric acid. The four cell types in the latter are neutrophil granulocytes, macrophages, eosinophil granulocytes, and basophil granulocytes. In addition to physical barriers and phagocytic cells mentioned above, antimicrobial proteins, such as complement proteins, acute phase proteins, and interferons, play an important role in further protecting the host.

While the adaptive immune system can respond to a diverse set of attackers (antigens), the innate immune system is limited to recognizing several common structures present in many different microorganisms. These innately recognizable structures are called pathogen-associated molecular patterns. Note that since there are a relatively small number of these patterns and they can be easily recognized by pattern-recognition receptors of most body defence cells, a response against foreign invaders is immediate.

11.3.2 Adaptive Immune System

During its lifetime an organism encounters numerous different antigens that the innate immune system is not able handle effectively. The adaptive or specific immune system comes into play in such a circumstance. Note that some of those antigens might have been previously exposed to the organism. In case of a subsequent invasion of the same antigen the adaptive immunity initiates a swifter and more effective response compared to the response to the first exposure to the invader. This is enabled by the memory function of the adaptive immune system. In addition to this memory property it functions with a series of sophisticated features, such as differentiation and self-organization. The adaptive immune system is often referred to simply as the immune system.

The two key components in the adaptive immune system are B lymphocytes (B cells) and T lymphocytes (T cells) of white blood cells that are produced by stem cells in the bone marrow. They are named after the lymphoid organs in which they are produced and developed, namely in the bone marrow and the thymus. Each of these two cell types plays a primary role in one of the two defence mechanisms (the humoral immunity and the cellular immunity).

The humoral immunity is overseen by B cells. More specifically, immunoglobulins or antibodies produced by plasma cells matured from B cells are the ones that actually take action. The humoral immune response takes place against invading microbes by antibody-antigen binding within a matter of minutes. As its name implies, the humoral immunity responses are activated in the body liquids, e.g., blood against bacteria and viruses. T cells, matured in the thymus as the primary mediator, take charge of the cellular immunity that responds to other cells that are infected by viruses. Now the question is how the adaptive immune system can respond against a virtually unlimited and diverse set of antigens. A sequence of phases for battling against these immunological enemies shown in Fig. 11.1 can answer this question.

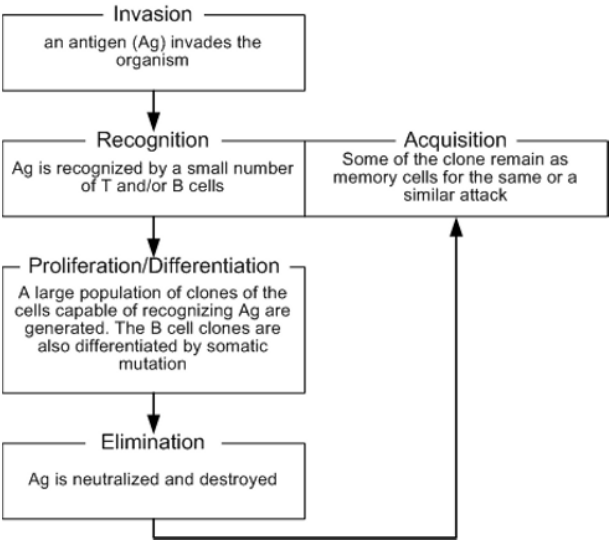


Fig. 11.1. The primary steps involved in the adaptive immune system.

11.3.3 Applicable Potentials of the Immune System: A Computational Science Perspective

The immune system consists of a complex, sophisticated, and effective set of properties, principles, and processes. There are a number of important immune features that should be noted, such as pattern recognition, adaptability, learning, and memory, to name a few. The key immune characteristics can be categorized into three types as follows:

- Self-organization and self-maintenance: The immune agents, primarily cells are autonomous in nature. Their activities, such as reaction to microbial attack, proliferation, differentiation, and memory take place without any central guidance or control. In essence, the immune system attempts to maintain superior cells while eliminating inferior and foreign cells. The key immune principle that enables and facilitates this feature is clonal selection with affinity maturation.
- Cooperativeness: There are two major sets of allies in the immune system—the innate and the adaptive immune systems, and T and B cells (or humoral and cellular immunity) in the adaptive immune system. While the latter allies are well known, the former are rarely recognized with fairly recent discoveries, such as dendritic cells and toll-like receptors as critical adjuvants in the activation of adaptive immunity. The cooperation and coordination of the two parties in each coalition exist for effective immune responses that no agent could achieve in isolation.
- Robustness: The immune system maintains a diverse set of lymphocytes distributed all over the body. These lymphocytes are constantly updated and/or upgraded. In this way immune responses can take place very quickly and effectively regardless of where the attack occurs or the nature of the attacking agent's pattern. In

addition, the immune system intensifies some lymphocytes upon the recognition of the attacker, in order to win the battle as soon and as easily as possible.

Note that the immune system does not rely on just one or two of these components. Rather, it works as an integrated system of all these powerful features. In other words, these characteristics should be carefully orchestrated when applied to computational problems.

11.4 Artificial Immune Systems

As the biological immune system is an effective defence against constant threats in dynamically changing environments, it has inspired models in an increasing number of areas, such as computer and network security, data mining, and pattern recognition. An artificial immune system can be defined as a real-world problem-solving methodology designed as a result of abstraction and modelling of immune features. For alternative definitions see de Castro and Timmis (2002).

To date the majority of AISs are based primarily upon the adaptive immune system mainly because it can deal with unforeseen circumstances. The two commonly modelled immune entities are antibodies and antigens, since they are the key players in the adaptive immune system. In conjunction with these immune entities a few immunological theories, such as negative/positive selection, clonal selection, and immune networks have been actively modelled. Note that these are merely some popularly modelled instances among a rich set of immune characteristics.

A recent analytical study (Garrett 2005) discusses how AISs differ from other biologically inspired computing approaches, such as EC, ANN, and FS. With its comparison study it evaluates the usefulness of AISs based on distinctiveness and effectiveness, and concludes that AISs have promising potential and have successfully demonstrated their applicability.

11.4.1 Negative Selection

The immune system's competence for recognizing foreign intruders is one very appealing function due to its direct applicability in many areas, such as anomaly detection and pattern recognition. Although there is not complete consensus as to how this recognition—so-called *self-nonself discrimination*—is accomplished, a single dominant mechanism of this immune activity is the negative selection principle.

It is crucial that the immune system does not become aggressive against its host. A general view of how the immune system distinguishes between self and nonself antigens is that there are only nonself recognizable T and B cells, as a result of the elimination of self-specific lymphocytes, circulating the body of a host. This selection takes place in the primary lymphoid organs of these lymphocytes before they are released.

Based upon this biological negative selection a well-known artificial negative selection scheme was proposed in Forrest et al. (1994), which modelled the negative

selection of T cells to detect changes in computer systems. The three principles of the algorithm presented in Forrest et al. (1994) take advantage of having a diverse set of change detectors that lead to the improvement of system robustness. Both self and nonself entities are represented as bit strings, namely self strings and nonself strings.

The negative selection algorithm consists of two major phases: detector generation and anomaly monitoring. In the detector generation phase, a set of detectors is randomly generated and matched against a predefined set of self strings. The detectors are filtered out based on how closely they match the self strings. Specifically, a preselected integer, r, is used as a censoring parameter that represents the number of contiguous matching bits. For example, if two strings have r or more contiguously identical bits they are called a closely matched pair. The detector of this pair then gets discarded due to its self-detecting ability. The anomaly monitoring phase continually matches between the detectors and the self strings in order to detect any changes in the self strings.

This flagship negative selection algorithm spawned a series of subsequent studies mainly in order to streamline the detector generation strategy and to widen its applicability. A number of improvements have been suggested to the random detector generation process, including linear and greedy algorithms (D'haeseleer et al. 1996), a deterministic analysis of negative selection (Wierzchon 2000), the permutation mask approach (Hofmeyr and Forrest 2000), and a detector generation scheme by random means with clonal selection support (Ayara et al. 2002).

Most applications of negative selection consider two opposing parties—good and evil. Here, the sole objective is to protect good from evil as in nearly every story and movie. Typical application areas include anomaly detection (Gonzalez et al. 2002; Gonzales and Cannady 2004), fault detection and diagnosis (Dasgupta et al. 2004; Gao et al. 2004), intrusion detection (Kim and Bentley 2001; Stibor et al. 2005), and more recently negative database (Esponda et al. 2005).

11.4.2 Danger Theory

For many years self-nonself discrimination has been widely regarded as the most compelling mechanism for immune responses. Traditionally, it is believed that an immune response is initiated upon the recognition of nonself antigens as in negative selection. However, this viewpoint was questioned especially by Matzinger (2002), who proposed a new immunological model—the danger theory. This model is grounded on the discrimination between dangerous and nondangerous instead of the conventional self-nonself classification. Burgess (1998) identified problems with the traditional model by posing the following questions: (1) Why does self-changing in the course of numerous lifetime events, such as puberty, pregnancy, and aging not trigger an immune response? (2) How are vaccines composed of inert foreign proteins harmoniously incorporated with the immune system? (3) What prevents the immune system from responding against those believed to be mutated proteins (e.g., tumors) or against autoreactive lymphocytes that might cause autoimmune diseases, such as Hashimoto's thyroiditis, Graves' disease, and rheumatoid arthritis?

These questions were answered by the danger model: the immune system responds not against any nonself, but against some that are recognized as a danger to the host. Then how do we define danger? In the biological immune system a cell can die in one of two ways: apoptosis or necrosis. The former is a programmed death as a result of homeostatic regulation, whereas the latter is a nonprogrammed death that might be due to a number of different causes, such as injury, infection, and inflammation. This cellular necrosis is believed to signal danger.

As well as the negative selection principle, danger theory has been applied to several similar areas (Burgess 1998; Aickelin and Cayzer 2002; Aickelin et al. 2003). The danger model is, however, still in an early stage of development. Aickelin and Cayzer (2002) and Aickelin et al. (2003) have introduced and described rather abstract models of its applications in anomaly detection and data mining, and intrusion detection, respectively.

An appealing strength of the danger theory for intrusion detection systems is its scalability. The most fundamental requirement for intrusion detection systems is that it define and model normal activities. Although negative selection is capable of this process based solely on the information of normal behaviours, it may suffer from a serious problem when the system is large and dynamic. In the meantime, the danger model is highly scalable since it only deals with activities known to be dangerous. Note that in this regard dangerous behaviours have to be defined, which gives rise to an adaptability issue.

11.4.3 Clonal Selection

As mentioned earlier, one of the most powerful features of the immune system is its adaptability. The clonal selection principle (Burnet 1959) in the adaptive immune system plays an important role in this property. Although clonal selection occurs on both T and B cells, the focus in the field of AIS is often aimed at B cells. This is primarily due to the fact that B cell clonal selection involves mutation that further enhances the adaptability of B cells. Hereafter, clonal selection simply refers to that of B cells.

The rationale behind the clonal selection theory is that superior B cells are preserved with a minor degree of mutation and become prevalent, whereas inferior ones are mutated at a high rate hoping to be improved and become rare. More specifically, when a foreign intruder (antigen) attacks the host, B cells matching the antigen will be cloned (i.e., clonal expansion) and mutated (i.e., affinity maturation) at rates directly and inversely proportional to the degree of the match (or affinity), respectively.

The clonal selection principle is the most popular model applied to AIS. In the past couple of decades a growing number of AIS (de Castro and Von Zuben 2002; Cutello and Nicosia 2002) based on the clonal selection theory have been developed. Some example applications can be found in areas of pattern recognition, scheduling, and graph coloring.

Among many existing AISs using this principle, CLONALG presented in de Castro and von Zuben (2002) is the most well known, largely due to its algorithmic simplicity. The algorithm CLONALG is described in Fig. 11.2.

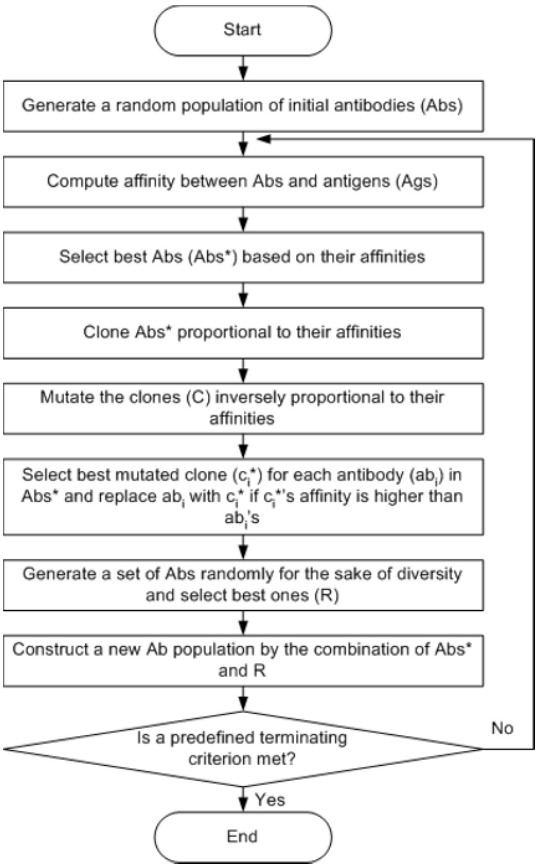


Fig. 11.2. The CLONALG algorithm.

11.4.4 Artificial Immune Networks

Another distinct immunological hypothesis attracting serious attention is Jerne's immune network theory (Jerne 1974). As its name implies the immune system is a network of self entities or antibodies that interact (i.e., stimulate and suppress) with each other as well as with foreign entities or antigens. As shown in Fig. 11.3 *paratopes* of an antibody bind not only to *epitopes* of an antigen, but also to *idiotopes* of other antibodies. Idiotopes—parts of antibodies—are the distinct feature introduced in the immune network theory. The existence of idiotopes contributes to establishing a network of antibodies; hence the name immune network theory or idiotypic network hypothesis. This leads to the assumption that the immune system operates dynamically even in the absence of antigen.

Like the clonal selection principle, superior antibodies proliferate and inferior antibodies are suppressed. That is, antibody clones with strong antigen and/or antibody recognition capability remain at high population levels, whereas less effective antibody

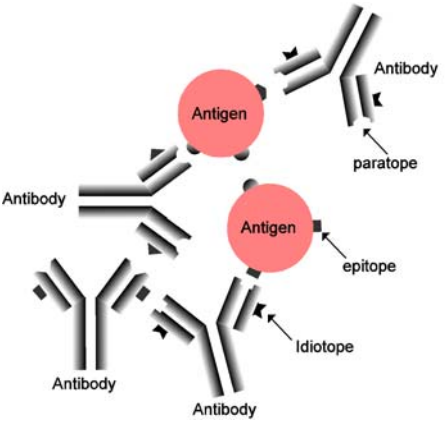


Fig. 11.3. The immune network theory.

clones are kept at lower population levels. This way, the immune system maintains the good-quality antibodies, resulting in effective immune responses. Once again the superiority measure used in the immune network theory is affinity, i.e., how strong is an antibody-antigen or antibody-antibody match.

Jerne's immune network theory initially populated several serious theoretical models (Farmer et al. 1986; Varela and Coutinho 1991) developed by immunologists which served as inspiration for computational network models for practical use. Two recent and well-known computational immune network models are RAIN (Timmis and Neal 2000) and AINET (de Castro and Von Zuben 2001). They both work similarly to the clonal selection principle, with one significant difference: the reactivity between antibodies. This is the most fundamental feature of immune network theory. More specifically, stimulatory and suppressive interactions between antibodies are incorporated in the process of regulating antibody clones based on affinities between them.

11.5 Abstraction and Modelling of the Immune System for Scheduling

A rich set of entities, principles, and processes in the immune system has shown great potential in many problem domains. It should be noted that successful exploitation of these immune features is heavily based on careful abstraction and modelling as in most, if not all, other nature-inspired approaches. Some examples of this abstraction and modelling in other approaches include artificial neurons in artificial neural networks, pheromones in ant colony optimization algorithms, and crossover and mutation processes in genetic algorithms.

Although different scheduling problem instances exhibit their own problem-specific properties they have a series of characteristics in common. For example, each task has processing amount and each resource is associated with processing capacity. The main objective of the scheduling problem, regardless of different problem

instances, is to find optimal task/resource allocation combinations taking into account certain performance metrics.

It should be noted that most AISs consist of a common set of steps: (1) initial population generation, (2) antibody-antigen binding, (3) population refinement, and (4) learning and adaptation. The last three steps iterate until a satisfactory solution is found.

11.5.1 Immune Entities

Both the scheduling problem and the immune system have two main entities—tasks or jobs and resources in the former and antibodies and antigens in the latter. Unlike the obvious mapping between the two immune entities and components in many problem domains, such as anomaly detection and pattern recognition, such mapping in scheduling is not obvious. Typically, most scheduling schemes based on the immune system use antibodies alone to represent schedules. However, other immune entities, such as antigens and lymphoid organs may also come into play in many scheduling problem instances.

Antibodies

As stated above, the most obvious immune component that can be modelled is the antibody. A typical representation of an antibody is a string of resource identifiers or simply resources, each with an associated task. Here, the arrangement of the resources determines the quality of the schedule.

Lymphoid Organs

As in the biological immune system the production of antibodies, or more precisely lymphocytes, in an AIS can be modelled using abstract lymphoid organs. The two primary lymphoid organs are the bone marrow and the thymus, and they can be abstracted and modelled for initial population generation and refinement, respectively.

In most AISs, the initial population is generated randomly. However, the use of a good-quality initial population might result in faster convergence with a better solution compared to that delivered using a randomly generated initial population.

The immunological crossover process (Bersini 2002) with a simple, yet efficient heuristic can be an option to generate decent-quality initial populations. More specifically, a heuristic is used to generate several solutions that are expected to be at least better than those generated randomly. The immunological crossover process further populates more initial solutions based on those generated using the heuristic.

In some particular scheduling problems in which similar or even the same sequences of tasks are regularly fed to a static set of resources, a schedule repository can be set up in order to store previous schedules. The repository maintains not only complete schedules, but also partial schedules commonly found in many complete schedules. Initial antibodies can be then composed via a process of schedule

assortment using the schedule repository. This approach resembles the actual antibody generation process of the immune system.

The initial population generated in an AIS may undergo a refinement process similar to the negative selection of lymphocytes (T and B cells). The process is used as an attempt to ensure that the antibodies or solutions in the initial population cover a wide range of search space. A simple way to refine the initial population is to remove an antibody that overlaps its certain portion with another one.

Antigens

Due to the fact that antigens tend not to present direct applicability to scheduling, their usefulness has been somewhat neglected. However, two possible applications of antigens are different task ordering and resource selection.

Typically, the order of tasks to be scheduled may be determined based on arrival time or priority, and it tends not to change in the course of scheduling. What is more, a single arrangement of the tasks is usually used. For example, tasks with precedence constraints in multiprocessor scheduling (e.g., list scheduling) are prioritized based on task dependency. There is a series of different prioritization methods, such as b-level, t-level, and s-level. Their priorities are used to determine the scheduling order. The tasks are then scheduled by using a task allocation scheme. Note that the scheduling order has an effect on the quality of the final schedule. Here, a number of different orders can play as antigens. An antibody that has a strong match (i.e., good schedule) with one or more of these antigens can then be selected.

Another possible use of antigens is the effective selection of resources. It is normally the case that the numbers of tasks and resources do not match perfectly. This implies that resources for the given tasks should be appropriately selected in order to generate a good, if not the optimal, schedule. In this regard an antigen can represent resource requirements, whereas an antibody represents available resources (e.g., memory and processing speed). An example of this usage (i.e., as a binary matching scheme) is introduced in King et al. (2001).

11.5.2 Immune Principles and Processes

An immune response is a result of a series of complex and sophisticated dynamics of immune entities. Over the lifetime of an organism these immune cells and molecules become increasingly more effective at responding to foreign intruders. This effectiveness is achieved by reinforcing the immune entities with various immune principles and processes. With an immune system based scheduling approach, one or more of these principles and processes typically are incorporated into it and carried out repeatedly, as in the biological immune system.

Negative Selection

The negative selection of lymphocytes (antibodies) can be modelled and adopted in a few processes of scheduling. The first two are, as described earlier, at the resource

selection and initial population generation stages. Another scheduling step in which the use of regulating self-reactive antibodies (similar to the negative selection process) can be found is at a late stage of scheduling schemes based on the immune network model. More specifically, in an immune-network-based scheduling approach the antibodies selected via a process of clonal selection are filtered out if they are bound by other antibodies, i.e., some schedules are nearly or exactly identical to others.

Danger Theory

The most fundamental issue in the danger theory is how to define and model a danger signal. The definition of a danger signal probably varies from one application to another. In dynamic scheduling the fluctuation of resource availability and capability can be modelled using danger theory. A danger signal in this case might be defined as a certain degree of deviation from the expected performance of a resource. It is most appropriate when the resource plays a critical role in the schedule.

If it is assumed that in a given system (e.g., a grid) schedules are first generated based on the static information of the resources and adapted during the actual scheduling process as the environment changes, then the failure or sudden overload of a resource can be considered as dangerous to the quality of the statically generated schedule resulting in modification of the schedule.

Clonal Selection

The clonal selection principle along with the affinity maturation process is the most popularly adopted and modelled immune feature for two reasons. Firstly the selection and mutation schemes are distinct and effective; they are not performed uniformly as in many evolutionary techniques, but directly and inversely proportional to the quality of a modelled immune entity, e.g., an antibody. Secondly, most major steps involved in an AIS based on the clonal selection principle are analogous to those found in other well-developed evolutionary techniques; hence they are easy to model and implement.

There have been a notable number of studies (King et al 2001; Costa et al. 2002; Coello Coello 2003; Ong et al. 2005) conducted on the clonal selection principle for various scheduling problems. The approaches proposed in these studies differ from one another mostly in terms of functions that determine the clonal selection rate and the hypermutation rate.

A clonal-selection-based scheduling approach typically models one immune entity (antibody) to represent schedules and two immune processes (clonal selection and affinity maturation) to enhance schedules ultimately aiming at finding the optimal schedule. The same principle as in the immune system applies: good schedules proliferate, whereas poor ones become extinct. These immune processes are the two primary sources of performance gain that AISs based on the clonal selection principle can exploit. The goodness of the former has a significant impact on narrowing down search space leading towards the optimal schedule. One may model it in one of the following two approaches, both based on antibody affinity (the quality of schedule):

- Proportional clonal generation. The clone size of an antibody is directly proportional to its affinity. The best clone among the clone set is compared with the original antibody and the better one is finally selected for the next generation.
- Proportional clonal selection. The number of clones for every antibody is fixed, whereas the number of clones for each antibody to be selected is directly proportional to the affinity of the antibody.

It is obvious that the higher the affinity of an antibody the more its clones are generated or selected.

Note that the final clonal selection process in both approaches takes place after the affinity maturation process to clone. More specifically, each clone of an antibody undergoes hypermutations and/or receptor editing as a process of differentiation hoping to improve its affinity. It is then evaluated based on its affinity.

An antibody with a high affinity value can be interpreted as a solution similar to the optimal one; hence fewer modifications compared to one with a low affinity value. There is a series of ways to mutate antibodies. A simple mutation method might be a random one that arbitrarily selects different points in an antibody and mutates them with randomly selected values. One can also adopt a mutation method that uses a set of systematic approaches to select the points to mutate and to generate new values. Here, the values of those mutating points might represent resource identifiers. A possible function that guides the hypermutation rate is the inverse of an exponential function (de Castro et al. 2002a).

The receptor editing process in the immune system is very similar to the negative selection process in the primary lymphoid organs except that entirely new antibodies are generated in place of those eliminated.

Immune Networks

One may see the immune network model as a superset of the clonal selection principle. In addition to those processes carried out with clones, artificial immune network (AIN) models involve antibody stimulation and suppression as given by immune network theory. These dynamics apply to the selected antibodies after the clonal selection process in order to further improve the superiority of the antibody population.

A simple application that models them with a slight modification in scheduling can be used to maintain the diversity as well as the quality of schedules. For example, some previously discarded schedules may be kept for a certain number of evolutions and compared with the current schedules. If any schedule in the current repertoire has been previously considered and discarded, it would be modified or replaced with a new one. A similar strategy can be found in TS.

11.5.3 Fitness Functions

As most immune-system-based scheduling approaches tend to be single player (antibody) games, commonly used distance metrics (e.g., the Euclidean distance and the Manhattan distance) in many AISs are adopted less frequently. Rather, the Hamming distance and schedule length (the most typical metric in scheduling) are widely accepted.

The Hamming distance between two strings is the number of symbols that are different. This metric can be used to measure the degree of matching in the resource selection process and the degree of schedule identity in negative selection and antibody stimulation and suppression in immune networks. Let S_1 and S_2 be two sets of schedules, each with a set of resources that are assigned to tasks. More formally,

$$S_1 = \{S_{1,1}, S_{1,2}, \ldots, S_{1,n}\} \text{ and } S_2 = \{S_{2,1}, S_{2,2}, \ldots, S_{2,n}\}.$$

The Hamming distance HD(S_1, S_2) between two schedules, S_1 and S_2 is

$$\text{HD}(S_1, S_2) = \sum_{i=1}^{n} \delta_i, \quad \text{where } \delta_i = \begin{cases} 0 & \text{if } S_{1,i} = S_{2,i} \\ 1 & \text{otherwise} \end{cases}. \tag{11.1}$$

The schedule length (the completion time) is defined as the amount of time from when the first task starts to when the last task finishes. The time complexity of a scheduling algorithm depends heavily on the computational cost of its fitness function. Thus, when modelling an immune-system-based scheduling algorithm one should carefully choose its parameters, such as the population size, the number of generations, and so on. This has a significant impact particularly on traditional multiprocessor scheduling, in that the amount of scheduling time one can afford is much less than that in other scheduling problems.

One use of the Euclidean distance is for identifying the similarity between a pair of antibodies in an immune network model (Zuo and Fan 2005). The Euclidean distance is given by

$$\text{ED}(S_1, S_2) = \sqrt{\sum_{i=1}^{n} (S_{1,i} - S_{2,i})}. \tag{11.2}$$

11.6 Scheduling Algorithms with Immune System Support: A Survey

The AIS approach has only recently drawn attention from researchers in the scheduling area. Although there are a growing number of AISs implementations for scheduling, many of them are not significantly different from each other. A selection of the unique AIS implementations is presented in this section.

11.6.1 Multiprocessor Scheduling

King et al. (2001) investigated functionalities of the immune system to design intelligent agents for task allocation in a heterogeneous computing environment. The main immune functionalities that inspired their AIS include the recognition process, learning, and memory mechanisms.

The AIS was designed with two intelligent agents, H cells and S cells, which control hardware resources and software properties and scheduling, respectively.

The H cells behave like typical resource managers in multiprocessor systems. However, they also use adaptive resonance theory (ART) as an adaptive method as well as the immunological functionalities mentioned above. Antigens are defined as adverse performance conditions and maintained by the H cells clustering them based on their similarity. This clustering facilitates antibody adaptation in that an antigen similar to those in a particular antigen cluster can be quickly identified.

The primary functionalities of the S cells are identifying the characteristics and resource requirements of a parallel program code and making scheduling decisions. In addition, they monitor the progress of program execution and perform rescheduling if any abnormal behaviour of the resources in the schedule is detected.

Costa et al. (2002) attempted to tackle an instance of the multiprocessor scheduling problem with the support of clonal selection and affinity maturation. Their AIS schedules a set of jobs to a set of identical parallel processors such that the completion time of the last processor to finish execution is the lowest possible. The AIS uses a lower-bound solution calculated by the sum of all job processing times divided by the number of processors with the use of preemption. Antibodies representing schedules (strings of processor IDs) are compared against this lower-bound solution to compute their affinity values. As usual the numbers of clones and mutations for each of the antibodies are determined by its affinity. Note that the number of mutations per antibody is empirically set. One interesting approach they used includes five types of mutation, and when a mutation is required, a mutation type is randomly selected.

11.6.2 Job Shop Scheduling

Coello Coello et al. (2003) applied the CLONALG algorithm with some modifications to approach the Job Shop Scheduling Problem (JSSP). In the JSSP, there are a set of jobs and a set of machines. Each job contains a series of a potentially different number of operations associated with precedence constraints and processing times. It is an NP-hard problem to assign the jobs onto the machines in such a way that the overall completion time of the jobs is minimal.

In each generation the AIS maintains an antigen and an antibody, where both are possible schedules. The antigen is initially a randomly generated schedule and is constantly updated with the best schedule in each of the following generations. While most AIS generate antibodies using a uniform random distribution, the antibody generation method in this AIS implementation is derived from the actual mechanism (concatenating gene segments) in the immune system.

The major steps of the AIS involved in each generation are as follows: (1) An antibody is generated and improved by a local search. (2) The antibody is compared with the antigen and it replaces the antigen if it is better than the antigen. (3) The antibody then gets cloned and mutated. (4) The best segments (i.e., the best job sequence for each machine) of the clone are stored in the gene libraries. Most parameters, such as the number of clones and the number of gene libraries used, are empirically determined.

Another AIS for the JSSP proposed is the adaptive scheduling system (Mori et al. 1998), which is based mainly on the immune network model. In addition to the min-

imization of makespan, the AIS attempts to achieve the minimization of setup and waiting times.

Antibodies are encoded in two different representations; hence two types. An antibody of type I is composed of a sequence of batch sizes, whereas one of type II represents a sequence of job priorities. The proliferation and suppression of antibodies is determined by the variety and the concentration of antibodies, respectively.

11.6.3 Flow Shop Scheduling

Engin and Doyen (2004) proposed an AIS based on the clonal selection principle of the immune system with some effort to optimize parameters to tackle hybrid flow shop (HFS) problems. The HFS problem in their study consists of a set J of jobs and a set P of machines. Each job in J is processed, undergoing a series of stages. At each stage it has to be processed on any one machine in P.

The proposed AIS modelled antibodies to represent sequences of jobs and the clonal selection principle with two different mutation methods: inverse and pairwise interchange. The size of the antibody population and the elimination ratio of antibodies in their AIS are the two parameters to be optimized. The rate of cloning for an antibody ab_i is calculated by a selection probability function defined by

$$CR(ab_i) = \frac{\max\limits_{ab_j \in AB} \{SL\,(ab_j)\} + 1 - SL\,(ab_i)}{\sum\limits_{j=1}^{|AB|} SL\,(ab_j)}, \tag{11.3}$$

where *AB* is the antibody population and *SL*(ab_i) is the makespan of the antibody ab_i. Apart from the adoption of a receptor editing process, the AIS basically works similarly to the CLONALG algorithm introduced earlier.

11.7 DAG Scheduling on Heterogeneous Computing Systems with Clonal Selection

This section presents an AIS for directed acyclic graph (DAG) scheduling in heterogeneous computing systems. The core immune component adopted for the AIS is the clonal selection principle. It is compared with a genetic algorithm (GA) and a well-known heuristic (HEFT) (Topcuoglu et al. 2002).

11.7.1 Problem Definition

Parallel programs, in general, can be represented by a directed acyclic graph. A DAG, $G = (V, E)$, consists of a set V of v nodes and a set E of e edges. A DAG is also called a task graph or macro-dataflow graph. In general, the nodes represent tasks partitioned from an application and the edges represent precedence constraints. An edge $(i, j) \in E$ between task n_i and task n_j also represents the intertask communication. In other

words, the output of task n_i has to be transmitted to task n_j in order for task n_j to start its execution. A task with no predecessors is called an *entry* task, n_{entry}, whereas an *exit* task, n_{exit}, is one that does not have any successors.

The target system used in this work consists of a set P of p heterogeneous processors/machines that are fully interconnected. The interprocessor communications are assumed to perform with the same speed on all links without contentions. It is also assumed that a message can be transmitted from one processor to another while a task is being executed on the recipient processor, which is possible in many systems.

The communication to computation ratio (CCR) is a measure that indicates whether a task graph is communication intensive or computation intensive. For a given task graph, it is computed by the average communication cost divided by the average computation cost on a target system.

The task scheduling problem in this study is the process of allocating a set P of p processors to a set V of v tasks aimed at minimizing SL without violating precedence constraints. The schedule length is defined as the maximum completion time of the exit task in a task graph.

11.7.2 The Proposed Artificial Immune System

The AIS presented in this section adopts antibodies and clonal selection for representing schedules and refining the quality of schedules, respectively. Its mutation method is distinct from those found in many other AISs in that mutations take place at idling slots of processors in a schedule. A set of randomly generated schedules is fed into the affinity maturation process of the AIS. Here, the incorporation of a randomly generated schedule into the original schedule (antibody) is regarded as a mutation. This can be viewed as 'schedule overlapping' resulting in duplicating tasks. The major benefit of this mutation scheme is the reduction of communication overhead. The AIS algorithm is presented in Algorithm 11.1.

Note that the AIS removes the redundant tasks (step 9) after each mutation. This is because some tasks, after the mutation, have no effect for the schedule.

The number of clones for the antibody ab_i and that of mutations for the clone c_i^j are determined by the following equations:

$$NC = max\{0,\ 2|AB| - |AB|\ ((SL(ab_i) - SSL)/SSL)\}, \tag{11.4}$$

$$NM = max\{2,\ 2|AB|\ ((SL(ab_i) - SSL)/SSL)\}, \tag{11.5}$$

where AB is the antibody population, $SL(ab_i)$ is the schedule length of the antibody ab_i, and SSL is the shortest schedule length among schedule lengths of all antibodies in AB.

11.7.3 Experiments and Results

Typically, the schedule length of a task graph generated by a scheduling algorithm is used as the main performance measure of the algorithm. The performance metric used for the comparison is the Normalized Schedule Length (NSL), which is defined as the

Algorithm 11.1: The proposed AIS.

```
1:  Generate initial antibody population at random
2:  for each generation do
3:      for each antibody ab_i in the antibody population AB do
4:          Clone ab_i proportional to its affinity (schedule length)
5:          for each clone c_i^j in the clone set C_i do
6:              Generate a set M of random schedules inversely proportional to ab_i's
                affinity
7:              for each random schedule m_k in M do
8:                  Mutate c_i^j with m_k
9:                  Remove redundant tasks in c_i^j
10:                 Compute the affinity of c_i^j
11:             end
12:         end
13:         Set ab_i to its best clone whose affinity is higher than ab_i's
14:     end
15:     Replace worst b% of antibodies in AB by randomly generated ones
16: end
```

schedule length obtained by a particular algorithm over that obtained by the HEFT algorithm.

The actual values of the parameters used for the AIS and GA are: (1) the number of generations = 10; (2) the numbers of antibodies/chromosomes = 10 and 55; (3) the number of mutations for GA = 10; and (4) the antibody elimination rate, $b = 20\%$.

The parameters used in the experiments are summarized in Table 11.1. The total number of experiments conducted with various randomly generated task graphs on the five different numbers of processors is 7200. More specifically, the random task graph set consists of 90 base task graphs generated with combinations of ten graph sizes, three CCRs and three processor heterogeneity settings. For each combination 20 task graphs are randomly generated retaining the base one's characteristics. These 1800 graphs are then experimented on with four different numbers of processors.

The computation and communication costs of the tasks in each task graph were randomly selected from a uniform distribution with the mean equal to the chosen average computation and communication costs. The processor heterogeneity value of 100

Table 11.1. Experimental parameters.

Parameter	Value
Number of tasks	U(10, 600)
CCR	{0.1, 1, 10}
Number of processors	{4, 8, 16, 32}
Processor heterogeneity	{100, 200, random}

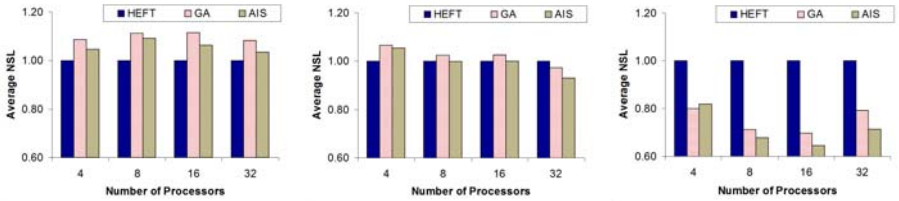


Fig. 11.4. Experimental results. Left: CCR = 0.1; centre: CCR = 1; right: CCR = 10.

is defined as the percentage of the speed difference between the fastest and the slowest processor in a given system.

It is clearly shown in Fig. 11.4 that the AIS delivers quite competitive schedule lengths irrespective of different application and system characteristics, e.g., graph sizes and the number of processors. The schedule lengths obtained from communication intensive task graphs indicate that the AIS best suits task graphs consisting of fine-grain tasks with large communication costs.

11.8 Conclusion

Although still in its infancy, the field of AIS has exhibited its potential in several areas including scheduling. The notable performance of AIS mostly benefits from the distinct adaptability and effectiveness of the adaptive immune system. A simple AIS for multiprocessor scheduling, presented in the previous section, once again demonstrates that AISs can be good alternatives for tackling many problems that are computationally hard and/or in dynamic environments.

Although an increasing number of researchers have been putting a lot of effort into applying immune features to various areas, many proposed AISs are limited to certain types of applications and to being developed based on a few popular principles and processes of the adaptive immune system. Therefore, the applicability of various other immune characteristics should be further investigated and exploited.

References

Aickelin, U., and Cayzer, S. (2002). The danger theory and its application to artificial immune systems. In Timmis, J. and Bentley, P. J., editors, *Proceedings of the First International Conference on Artificial Immune Systems (ICARIS)*, pages 141–148. University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.

Aickelin, U., Bentley, P., Cayzer, S., Kim, J., and McLeod, J. (2003). Danger theory: The link between AIS and IDS? *Proceedings of the Second International Conference on Artificial Immune Systems (ICARIS)*, volume 2787 of *Lecture Notes in Computer Science*, pages 147–155. Edinburgh, UK, Springer.

Ayara, M., Timmis, J., de Lemos, R., de Castro, L., and Duncan, R. (2002). Negative selection: How to generate detectors. In Timmis, J., and Bentley, P. J., editors, *Proceedings of the First*

International Conference on Artificial Immune Systems (ICARIS), pages 89–98. University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.

Bersini, H. (2002). The immune and the chemical crossover. *IEEE Transactions on Evolutionary Computation*, 6(3):306–313.

Burgess, M. (1998). Computer immunology. *Proceedings of the 12th USENIX Conference on System Administration*, pages 283–298. Boston, Massachusetts, USA, USENIX Association.

Burnet, F. M. (1959). *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press. Cambridge.

Coello Coello, C.A., Rivera, D. C. and Cortés, N. C. (2003). Use of an artificial immune system for job shop scheduling. *Proceedings of the Second International Conference on Artificial Immune Systems (ICARIS)*, volume 2787/2003 of *Lecture Notes in Computer Science*, pages 1–10. Edinburgh, UK, Springer.

Costa, A. M., Vargas, P. A., Von Zuben, F. J. and Franca, P. M. (2002). Makespan minimization on parallel processors: An immune-based approach. *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, volume 1, pages 920–925. IEEE Computer Society, Washington DC, USA.

Cutello, V. and Nicosia, G. (2002). Multiple learning using immune algorithms. In *Fourth International Conference on Recent Advances in Soft Computing (RASC-2002)*, pages 102-107. Nottingham. Springer-Verlag, Berlin.

D'haeseleer, P., Forrest, S., and Helman, P. (1996). An immunological approach to change detection: Algorithms, analysis and implications. *Proceedings of IEEE Symposium on Security and Privacy,* pages 132–143. Oakland, CA, USA, IEEE Computer Society.

Dasgupta, D., Cao, Y. and Yang, C. (1999). An immunogenetic approach to spectra recognition. In: Banzhaf, W., Daida, J. M., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M. J., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 149–155. Morgan Kaufmann, San Fransisco, LA, USA.

Dasgupta, D., KrishnaKumar, K., Wong, D. and Berry, M. (2004). Negative selection algorithm for aircraft fault detection. *Proceedings of the Third International Conference on Artificial Immune Systems (ICARIS)*, volume 3239 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin.

de Castro, L. N., and Timmis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, London.

de Castro, L. N., and Von Zuben, F. J. (2001). AINET: An artificial immune network for data analysis. In Abbass, H. A., Sarker, R. A., and Newton, C. S., editors, *Data Mining: A Heuristic Approach*, chapter XII, pages 231–259. Idea Group Publishing Hershey, PA, USA.

de Castro, L. N., and Von Zuben, F. J. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251.

Engin, O., and Doyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generation Computer Systems*, 20(6):1083–1095.

Esponda, F., Ackley, E. S., Forrest, S. and Helman, P. (2005). On-line negative databases. *International Journal of Unconventional Computing*, 1(3):201–220.

Farmer, J., Packard, N., and Perelson, A. (1986). The immune system, adaptation and machine learning. *Physica D*, 22:187–204.

Feng, Y.-J., and Feng, Z.-R. (2004). An immunity-based ant system for continuous space multimodal function optimization. *Proceedings of International Conference on Machine Learning and Cybernetics*, volume 2, pages 1050–1054. IEEE Computer Society, Washington DC, USA.

Forrest, S., Perelson, A. S., Allen, L. and Cherukuri, R. (1994). Self-nonself discrimination in a computer. *Proceedings of IEEE Symposium Research in Security and Privacy*, pages 202–212. IEEE Computer Society, Washington DC, USA.

Gao, X.Z., Ovaska, S.J., Wang, X. and Chow, M.-Y. (2004). Neural networks-based negative selection algorithm with applications in fault diagnosis. *Proceedings of International Conference on Systems, Man and Cybernetics,* volume 4, pages 3408–3414.

Garrett, S. (2005). How Do We Evaluate Artificial Immune Systems?. *Evolutionary Computation*, 13(2):145–177.

Gonzales, L.J., and Cannady, J. (2004). A self-adaptive negative selection approach for anomaly detection. *Proceedings of Congress on Evolutionary Computation (CEC 04),* volume 2, pages 1561–1568.

Gonzalez, F., Dasgupta, D. and Kozma, R. (2002). Combining negative selection and classification techniques for anomaly detection. *Proceedings of Congress on Evolutionary Computation (CEC'02)*, volume 1, pages 705–710.

Grama, A., Gupta, A., Karypis, G. and Kumar, V. (2003). *Introduction to Parallel Computing*. 2nd Edition, Addison Wesley, Boston, MA, USA.

Hajela, P., and Yoo. J. S. (1999). Immune network modelling in design optimization. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 203–215. McGraw Hill, London.

Hofmeyr, S., and Forrest, S. (2000). Architecture for the artificial immune system. *Evolutionary Computation*, 8(4):443–473.

Jerne, N. (1974). Towards a network theory of the immune system. *Annals of Immunology*, 125:373–389.

Kim, J., and Bentley, P.J. (2001). Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. *Proceedings of Congress on Evolutionary Computation (CEC'01),* volume 2, pages 1244–1252. IEEE Computer Society, Washington DC, USA.

King, R. L., Russ, S. H., Lambert, A. B., and Reese, D. S. (2001). An artificial immune system model for intelligent agents. *Future Generation Computer Systems*, 17(4):335–343.

KrishnaKumar, K., Satyadas, A. and Neidhoefer, J. (1995). An immune system framework for integrating computational intelligence paradigms with applications to adaptive control. In Palaniswami, M., Attikiouzel, Y., Marks, R.J., II, Fogel, D., and Fukuda T., editors, *Computational Intelligence A Dynamic System Perspective*, pages 32–45, IEEE.

Kwok, Y. K., and Ahmad, I. (1998). Benchmarking the task graph scheduling algorithms. *Proceedings of First Merged International Parallel Symposium/Symposium on Parallel and Distributed Processing (IPPS/SPDP '98)*, pages 531–537. IEEE Computer Society, Washington DC, USA.

Matzinger, P. (2002). The danger model: A renewed sense of self. *Science*, 296:301–305.

Mori, K., Tsukiyama, M. and Fukuda, T. (1998). Adaptive scheduling system inspired by immune system. *Proceedings of International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3833–3837. IEEE Computer Society, Washington DC, USA.

Ong, Z. X, Tay, J. C. and Kwoh, C. K. (2005). Applying the clonal selection principle to find flexible job-shop schedules. *Proceedings of International Conference on Artificial Immune Systems (ICARIS)*, pages 442–455, Springer-Verlag Berlin.

Stibor, T., Timmis, J. and Eckert, C. (2005). On the appropriateness of negative selection defined over Hamming shape-space as a network intrusion detection system. *Proceedings of Congress on Evolutionary Computation*, volume 2, pages 995–002. IEEE Computer Society, Washington DC, USA.

Swiecicka, A., Seredynski, F., and Zomaya, A.Y. (2006). Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support. *IEEE Transactions on Parallel and Distributed Systems*, 17(3):253–262.

Timmis, J., and Neal, M. J. (2000). A resource limited artificial immune system for data analysis. *Proceedings of ES 2000*, pages 19–32, Springer.

Topcuoglu, H., Hariri, S. and Wu, M. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274.

Varela, F. J., and Coutinho, A. (1991). Second generation immune networks. *Immunology Today*, 12(55):159–166.

Wierzchon, S. T. (2000). Discriminative power of the receptors activated by k-contiguous bits rule. *Journal of Computer Science and Technology, Special Issue on Research Computer Science*, 1(3):1–13.

Zuo, X.-Q., and Fan, Y.-S. (2005). Solving the job shop scheduling problem by an immune algorithm. *Proceedings of International Conference on Machine Learning and Cybernetics*, volume 6, pages 3282–3287. IEEE Computer Society, Washington DC, USA.

12

Formal Immune Networks: Self-Organization and Real-World Applications

Alexander O. Tarakanov

12.1 Introduction

Two types of self-organizing biological systems in vertebrates—the neural system and the immune system—possess the capabilities for "intelligent" information processing, which include memory and the ability to learn, to recognize, and to make decisions with respect to unknown situations. The potential of the natural neural system as a biological prototype of a computing scheme has already been well established as a field of artificial neural networks, or neural computing (Cloete and Zurada 2000). However, the computing capabilities of the natural immune system (Jerne 1973, 1974; de Boer et al. 1992) have only recently begun to be appreciated as a field of artificial immune systems (AIS) (Dasgupta 1999; de Castro and Timmis 2002; NASA 2004). The mathematical formalization of these capabilities (Tarakanov and Dasgupta 2000) forms the basis of immunocomputing (IC) as a new approach that replicates the principles of information processing by proteins and immune networks (Tarakanov et al. 2003).

The IC approach appears to have good potential as a basis for a new kind of computing. A series of successful applications of IC to real-world tasks, has been reported, including detection of dangerous ballistic situations in near-Earth space (Tarakanov and Dasgupta 2002), computing of ecological maps, and optical response of laser diodes (Tarakanov and Tarakanov 2004, 2005), and reconstruction of hydroacoustic fields (Tarakanov et al. 2007). It is also worth noting that a combination of IC and cellular automata has led to encouraging results in three-dimensional (3D) computer graphics (Tarakanov and Adamatzky 2002).

As for biological applications of IC, the concept of a "biomolecular immunocomputer" as a computer-controlled fragment of the natural immune system has recently been reported (Goncharova et al. 2005). The use of IC in connection with brain research has also led to promising results in understanding of basic principles of the organization of receptor mosaics and molecular networks (Agnati et al. 2005a,b).

It is against such a background, that this chapter proposes a generalized model of a formal immune network (FIN) based on self-organizing features of apoptosis (programmed cell death) and autoimmunization, both induced by cytokines (messenger proteins). The chapter also describes real-world applications of FIN to intrusion

detection in computer networks and forecasts of hydrophysical fields in the ocean. The obtained results demonstrate that FIN outperforms (by training time and accuracy) other approaches to computational intelligence as well as more conventional methods of interpolation.

12.2 Biomolecular Background

Cytokines are a group of biologically active mediator molecules that signal intercellular interactions within the immune system. They are the central regulators of leukocyte growth and differentiation, being produced by a wide variety of cell types, targeting various cell subsets, and exhibiting numerous biological activities. More than 100 different human cytokines have already been identified, and an increasing volume of experimental data suggests that cytokines play a central role in immune regulation as well as in neuro-immune-endocrine modulation (Ader et al. 2001). The notion of cytokines as a network modulating and switching several cascades of immune reactions (Balkwill 2000) adjoins with the concept that considers such molecules as a field or a milieu with local properties that mediate immune response (Kourilsky and Truffa-Bachi 2001).

A relationship exists between cytokine levels in human body fluids and disease pathogenesis, including inflammation and even depression (Bunk 2003). Many types of cancers interfere with the regulatory role of cytokines to down-regulate appropriate immune responses targeted at destroying cancer cells by secreting immunosuppressive cytokines that induce generalized and specific inhibition of immune responses (Kurzrock 2001; Igney and Krammer 2002). Thus, the use of immunostimulatory cytokines as tumor vaccines is a potentially promising strategy in cancer immunotherapy (Vilcek and Feldman 2004).

Recent developments show that cytokines induce apoptosis in cancer cells (Wall et al. 2003). The induction of apoptosis is associated with a dose-dependent inhibition of cancer cell division, and this activity has been demonstrated for a wide range of cancer types including bladder, breast, leukemia, melanoma, ovarian, and prostate.

Apoptosis is a natural mechanism by which cells "commit suicide" when they have outlived their purpose, become defective, or have aged, thus preventing cells from accumulating and forming tumors. Understanding of the control of apoptosis in normal and malignant cells will help to improve the diagnosis and treatment of malignancies. The goal of many treatments, including chemotherapies, is to induce malignant cells to undergo apoptosis. Current data also suggest that a cytokine may function as a dual-acting cytokine in which its normal physiological functions may be related to specific aspects of the immune system and overexpression culminates in cancer-specific apoptosis (Fisher et al. 2003).

Based on such a biomolecular background, we previously proposed the idea of cytokine FIN (cFIN) (Tarakanov et al. 2005a). In the following pages this mathematical notion is generalized and applied to two tasks where self-organizing features of FIN seem to play a key role.

12.3 General Mathematical Model

Vector-matrix transposing is designated an apostrophe $[]'$. For example, if X is a column vector, then X' is a row vector. The end of a proof is designated by the symbol $\diamondsuit$ (rhomb).

Definition 1. A cell is a pair *V = (f, P)*, where f is a real value, $f \in R$, and $P = (p_1, ..., p_q)$ is a point in q-dimensional space: $P \in R^q$, and P lies within a unit cube: $\max\{|p_1|, ..., |p_q|\} \leq 1$.

Let distance (affinity) $d_{ij} = d(V_i, V_j)$ between cells V_i and V_j be defined by a norm:

$$d_{ij} = \|P_i - P_j\|,$$

which can be Euclidian $\|P\|_E$, Manhattan $\|P\|_M$, Tchebyshev $\|P\|_T$, or any other appropriate norm:

$$\|P\|_E = \sqrt{p_1^2 + \ldots + p_q^2},$$
$$\|P\|_M = |p_1| + \ldots + |p_q|,$$
$$\|P\|_T = \max\{|p_1|, \ldots, |p_q|\}.$$

Fix some finite nonempty set of cells (innate immunity) $W_0 = (V_1, ..., V_m)$ with nonzero distance between cells: $d_{ij} \neq 0, \forall i, j$: $i \neq j$.

Definition 2. FIN is a set of cells: $W \subseteq W_0$.

Definition 3. Cell V_i recognizes cell V_k if the following conditions are satisfied:

$$|f_i - f_k| < \rho, \quad d_{ik} < h, \quad d_{ik} < d_{ij}, \quad \forall V_j \in W, \quad j \neq i, \quad k \neq j,$$

where $\rho \geq 0$ and $h \geq 0$ are nonnegative real values (recognition threshold and affinity threshold).

Let us define the behavior (self-organization) of FIN by the following two rules.

- *Rule 1* (apoptosis). If cell $V_i \in W$ recognizes cell $V_k \in W$ then remove V_i from FIN.
- *Rule 2* (autoimmunization). If cell $V_k \in W$ is nearest to cell $V_i \in W_0 \backslash W$ among all the cells of FIN: $d_{ik} < d_{ij}, \ \forall V_j \in W$, whereas if $|f_i - f_k| \geq \rho$, then add V_i to FIN.

Let W_A be FIN as a consequence of the application of apoptosis to all cells of W_0. Let W_I be FIN as a consequence of the autoimmunization of all cells of W_A by all cells of W_0. Note that the resulting sets W_A and W_I depend on the ordering of cells in W_0. Further it will be assumed that the ordering is given.

Let us consider the general mathematical properties of FIN. It is obvious that neither the result of apoptosis W_A nor the result of autoimmunization W_I can overcome W_0 for any innate immunity or threshold:

$$W_A \subseteq W_0, \quad W_I \subseteq W_0, \quad \forall W_0, \, h, \, \rho.$$

The following propositions give more important and less evident properties of FIN.

Proposition 1. For any innate immunity W_0 and recognition threshold ρ there exists an affinity threshold h_0 such that apoptosis does not change W_0 for any h less than h_0: $W_A = W_0, \forall h < h_0$.

Proof. Let h_0 be the minimal distance for any pair of FIN cells that satisfy the recognition threshold:

$$h_0 = \min_{i,j}\{d_{ij}\}: \quad |f_i - f_j| < \rho, \quad i \neq j.$$

Then, according to Definition 3, none of the FIN cells can recognize other cells, because $d_{ij} > h_0$ for any pair of cells V_i and V_j. According to Rule 1, none of the cells can be removed from FIN for any h less than h_0, because $d_{ij} > h$, $\forall h < h_0$, $\forall V_i, V_j \in W_0$. Thus, $W_A = W_0$, $\forall h < h_0$. $\diamondsuit$

Proposition 2. For any innate immunity W_0 and recognition threshold ρ there exists an affinity threshold h_1 such that the consequence of apoptosis and autoimmunization $W_1 = W_I(h_1)$ provides the minimal number of cells $|W_1|$ for given W_0 and ρ, and any h: $|W_1| \leq |W_I(h)|$, $\forall h$, $\forall W_I \subseteq W_0$.

Proof. Let h_1 be the maximal distance for any pair of FIN cells that satisfy the recognition threshold:

$$h_1 = \max_{i,j}\{d_{ij}\}: \quad |f_i - f_j| < \rho, \quad i \neq j.$$

Then, according to Definition 3, any cell V_i can recognize the nearest cell V_j if the latter satisfies the recognition threshold: $|f_i - f_j| < \rho$. Let W_- be the set of all such cells V_i. Then, according to Rule 1, $|W_A(h_1)| = |W_0| - |W_-|$, and the number of such cells after apoptosis is minimal among any h: $|W_A(h_1)| \leq |W_A(h)|$, $\forall h$.

Let W_+ be a set of cells that is added to $W_A(h_1)$ as a consequence of autoimmunization: $W_1 = W_A(h_1) \cup W_+$. It is also evident that W_+ is a subset of W_-: $W_+ \subseteq W_-$, and $|W_+|$ represents a number of "mistakes" of apoptosis when FIN "kills" some cells, which leads to further recognition errors. Such cells are then "restored" by autoimmunization (Rule 2).

Let $W_* = W_- \backslash W_+$ be cells that yield apoptosis without further recognition errors. Then $|W_+| = |W_-| - |W_*|$. On the other hand, $|W_1| = |W_A(h_1)| + |W_+|$. Substitutions of $|W_A(h_1)|$ and $|W_+|$ lead to the following result: $|W_1| = |W_0| - |W_*|$. Thus, $|W_1| \leq |W_I(h)|$, which proves Proposition 2. $\diamondsuit$

Actually, Proposition 2 states that the minimal number of cells after apoptosis and autoimmunization is a kind of "inner invariant" of any FIN, which depends on the innate immunity and the recognition threshold but not on the affinity threshold. Practically, it means that such an invariant can be found for any FIN by apoptosis and autoimmunization without considering any affinity threshold (in Definition 3) at all.

Now we can define a general model of molecular recognition in terms of FIN. Let the epitope (antigenic determinant) be any point $P = (p_1, ..., p_q)$ of q-dimensional space: $P \in R^q$. Note that any cell of FIN also contains an epitope, according to Definition 1.

Definition 4. Cell V_i recognizes epitope P by assigning it the value f_i if the distance $d(V_i, P)$ between the cell and the epitope is minimal among all the FIN cells: $d(V_i, P) = \min\{d(V_j, P)\}$, $\forall V_j \in W$.

Let pattern (molecule) be any n-dimensional column vector $Z = [z_1, ..., z_n]'$, where $z_1, ..., z_n$ are real values. Let pattern recognition be mapping of the pattern to an epitope: $Z \rightarrow P$, and recognition of the epitope by the value f of the nearest FIN cell.

Let $X_1, ..., X_m$ be a set of n-dimensional patterns (cells) with known values $f_1, ..., f_m$. Let $A = [X_1 \ ... \ X_m]'$ be a matrix of dimension $m \times n$. Consider singular value decomposition (SVD) of this matrix (Horn and Johnson 1986):

$$A = s_1 L_1 R_1' + ... + s_q L_q R_q' + ... + s_r L_r R_r', \tag{12.1}$$

where r is the rank of matrix A, s_k are singular values, and L_k and R_k are left and right singular vectors with the following properties:

$$L_k' L_k = 1,\ R_k' R_k = 1,\ L_k' L_i = 0,\ R_k' R_i = 0,\ i \neq k,\ k = 1, ..., r,\ s_{k-1} \geq s_k, k > 1.$$

Consider the following mapping of any n-dimensional pattern Z to epitope P:

$$p_k = \frac{1}{s_k} Z' R_k, \quad k = 1, ..., q, \quad q \leq r. \tag{12.2}$$

Note that any epitope obtained by application of Eq. (12.2) to any training pattern lies within a unit cube (see Definition 1), according to the above properties of singular vectors.

If value f (in Definition 1) is a natural number $f = c$, where $c \in N$ (i.e., cytokine or class), whereas the recognition threshold (in Definition 3) $\rho < 1$ and distance between cells is determined by the Tchebyshev norm, then we obtain a special case of cFIN (see Sec. 12.2) proposed by Tarakanov et al. (2005a) and applied to discrete pattern recognition.

12.4 General Pattern Recognition Algorithm

The IC approach to pattern recognition is inspired by the principle of molecular recognition between proteins, including an antibody (also called an immunoglobulin) of the natural immune system and any other antigen (including another antibody). Consider the following informal example to shed light on this concept.

Let *Ig1* and *Ig2* be two antibodies and *Ag* be an antigen. The strength of biophysical interaction between any pair of proteins can be measured by their binding energy. Let $e1, e2$ be values of the binding energy between *Ag* and *Ig1, Ig2*, respectively. Then any protein (including an antibody) can be presented and recognized by the corresponding pair of numbers $e1$ and $e2$ in a two-dimensional immune network of interactions formed by two antibodies *Ig1* and *Ig2*. A more formal description of this idea is as follows.

In terms of a general model (see previous section), any n-dimensional input vector Z ("antigen") is projected to the q-dimensional space of FIN and recognized by class (value f, in general) of the nearest point (cell) of FIN (e.g., see Fig. 12.1, where $q = 2$). Coordinate axes of such FIN spaces are determined by right singular vectors

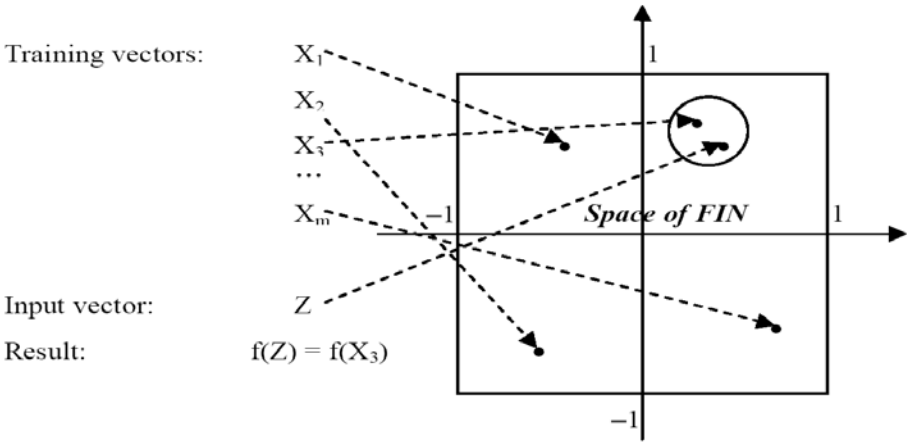


Fig. 12.1. Example of pattern recognition in a FIN 2D space.

("antibodies") of SVD of the training matrix $A = [X_1...X_m]'$, where $X_1, ..., X_m$ are n-dimensional training vectors.

Using SVD to construct antibodies of FIN has some theoretical and practical advantages over other methods of extracting features from training data. For example, SVD guarantees mathematically optimal reconstruction of the training matrix by a reduced set of components so that least-square error is minimal compared to all other methods. As a method of linear algebra, SVD can be implemented by a relatively simple and robust algorithm.

In line with the general model of FIN, consider the following description (in pseudocode) of a generalized IC algorithm of pattern recognition that provides real-valued (continuous) output f for any input vector Z. The main idea of this generalization is to find more than one nearest cell (point) in the FIN space and interpolate the FIN output using the known values of the function in these nearest training points.
Steps 1–7 below describe the algorithm in more rigorous mathematical terms. Note that Steps 1–5 are also usual for the IC algorithm of (discrete) pattern recognition, whereas Steps 6 and 7 provide a real-valued (continuous) output of the generalized FIN.

1. Form training matrix $A = [X_1...X_m]'$ of dimension $m \times n$.
2. Compute first q singular values $s_1, ..., s_q$ and corresponding left and right singular vectors $L_1, ..., L_q$ and $R_1, ..., R_q$ by SVD of training matrix (12.1), where $q \leq r$ and r is the rank of the matrix.
   According to Tarakanov et al. (2003), such SVD can be computed by the following iterative scheme (Steps 2.1–2.3).
   2.1. Compute the maximal singular value s_1 and the corresponding singular vectors L_1 and R_1 of the training matrix:

$$L_{(0)} = [1 \ \ldots \ 1]',$$

Algorithm 12.1: Generalized IC algorithm of pattern recognition.

```
1:  while Training do
2:      begin 1st stage training            // map training data to FIN
3:          Get training patterns
4:          Form training matrix
5:          Compute SVD of the training matrix
6:          Store q singular values         // "binding energies"
7:          Store q right singular vectors  // "antibodies"
8:          Store left singular vectors     // cells (points) of FIN
9:      end
10:     begin 2nd stage training
11:         // compress data by "self-organization" of FIN
12:         // compute consecutively for all cells of FIN:
13:         begin Apoptosis
14:             if cell[i1] is the nearest to cell[i2] and
                abs(f.cell[i1]-f.cell[i2])<recognition_threshold then
15:                 kill cell[i1]
16:             end
17:         end
18:         begin Autoimmunization      // correct mistakes of Apoptosis
19:             if cell[i1] is the nearest to cell[i2] and
                abs(f.cell[i1]-f.cell[i2])>=recognition_threshold then
20:                 restore cell[i1]
21:             end
22:         end
23:     end
24: end
25: while Recognition do
26:     Get pattern                        // "antigen"
27:     Map the pattern to FIN
28:     Find p nearest cells of FIN
29:     Interpolate value f by the values of p nearest cells
30:     Assign the interpolated value to the pattern
31: end
```

$$R' = L'_{(k-1)}A, \quad R_{(k)} = \frac{R}{|R|},$$

$$L = AR_{(k)}, \quad L_{(k)} = \frac{L}{|L|},$$

$$s_{(k)} = L'_{(k)}AR_{(k)},$$

where $|X| = \|X\|_E$ and $k = 1, 2, \ldots$ until the following condition is satisfied for given constant ε:

$$\left|s_{(k)} - s_{(k-1)}\right| < \varepsilon.$$

Then

$$s_1 = s_{(k)}, \quad L_1 = L_{(k)}, \quad R_1 = R_{(k)}.$$

2.2. Form matrices:

$$A_2 = A - s_1 L_1 R_1', \;\; A_3 = A_2 - s_2 L_2 R_2', \;\; \ldots, \;\; A_q = A_{q-1} - s_{q-1} L_{q-1} R_{q-1}',$$

and compute their maximal singular values and corresponding singular vectors by Step 2.1.

2.3. Store q singular values $s_1, \ldots, s_q$ and right and left singular vectors $R_1, \ldots, R_q$ and $L_1, \ldots, L_q$.

3. For any training vector X_i, compute its mapping $Y(X_i) = [y_{i1} ... y_{iq}]'$ to FIN q-dimensional space:

$$y_{i1} = \frac{1}{s_1} X_i' R_1, \ldots, \quad y_{iq} = \frac{1}{s_q} X_i' R_q.$$

4. Use apoptosis and autoimmunization to reduce m FIN training points to $k \leq m$ points, where the number of points k is "self-defined" by the inner invariant of FIN (see Proposition 2).
5. For any n-dimensional vector Z, compute its mapping $Y(Z) = [y_1 ... y_q]'$ to FIN q-dimensional space:

$$y_1 = \frac{1}{s_1} Z' R_1, \ldots, \quad y_q = \frac{1}{s_q} Z' R_q.$$

6. Among the reduced FIN training points $Y_1, \ldots, Y_k$, determine p nearest to $Y(Z)$ points $Y_1, ..., Y_p$ and their distances:

$$d_1 = \|Y_1 - Y\|, \ldots, \quad d_p = \|Y_p - Y\|.$$

7. Interpolate $f(Z)$ by the following sum:

$$f = \sum_{i=1}^{p} a_i f_i,$$

where $f_i = f(Y_i)$ are training values that correspond to the nearest points of FIN, whereas coefficients a_i are determined by the distances:

$$a_i = \frac{1}{1 + d_i \sum\limits_{j \neq i}^{p} (1/d_j)}.$$

Note the following useful features of FIN. It can be shown that

$$\sum_{i=1}^{p} a_i = 1.$$

It can be also shown that $f = f_i$ if $d_i = 0$ for any i (then $d_j \neq 0$ for any $j \neq i$). To prove this, consider a special case when the input antigen represents a row of the

training matrix and, thus, is equal to a training vector: $Z = X_i, \quad i = 1, \ldots, m$. According to SVD properties utilized in Steps 1–3, the projection of such an antigen to the FIN space coincides with the corresponding FIN training point: $Y(Z) = Y(X_i)$. In such a case, Step 4 calculates the following distances of the nearest FIN points: $d_1 = 0, d_2 \neq 0, \ \ldots, \ d_p \neq 0$. Then, according to Step 7: $a_1 = 1, a_2 = 0, \ \ldots \ a_p = 0$, and the output of FIN is equal to the value of the function $f(X_i)$ for the corresponding training vector X_i: $f = f_i$. Thus, IC does not make mistakes on any training set.

Note that Step 2 can also be considered as an example of FIN self-organization (together with the apoptosis and autoimmunization of Step 4). It can be said that iterations of Step 2 for any training set "self-converge" to antibodies. More rigorously, such convergence can be derived from the Rayleigh-Ritz theorem (Horn and Johnson 1986) due to the fact that the maximal singular value of any matrix A is actually the maximum of the bilinear form $L'AR$ over unit vectors $L'L = 1, \ R'R = 1$.

It is also worth noting that this algorithm can be supplied by on-line learning capabilities. In the case of a change in any training vector X_i, one can just compute its mapping to the FIN space (by Step 3) and add this point to the reduced FIN training points (to use in Step 6). Therefore, in case of a change in the training patterns, the training phase does not necessarily have to be repeated with the new training set.

12.5 Intrusion Detection in Computer Networks

A special case of cFIN (see Secs. 12.2 and 12.3) has been implemented as a version of the "immunochip emulator" (Tarakanov et al. 2005b) using Visual C++ with a built-in assembler code of the affinity function (Tchebyshev norm) in 3D space ($q = 3$) and OpenGL tools for user friendly 3D visualization. A screen shot of the emulator is shown in Fig. 12.2.

The known UCI KDD archive (Bay 1999) has been used for testing the emulator. Lincoln Labs set up an environment to acquire nine weeks of raw TCP (transmission control protocol) dump data simulating a typical US Air Force local area network (LAN). They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about 4 gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

A connection is a sequence of TCP packets starting and ending at some well-defined time, between which data flow to and from a source IP (internet protocol) address to a target IP address under some well-defined protocol. Each connection is labeled as either normal or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

Two data files from the KDD archive were used to test the emulator:

- File 1: kddcup_data_10_percent_gz.htm (7.7 MB);
- File 2: kddcup_newtestdata_10_percent_unlabeled_gz.htm (44 MB).

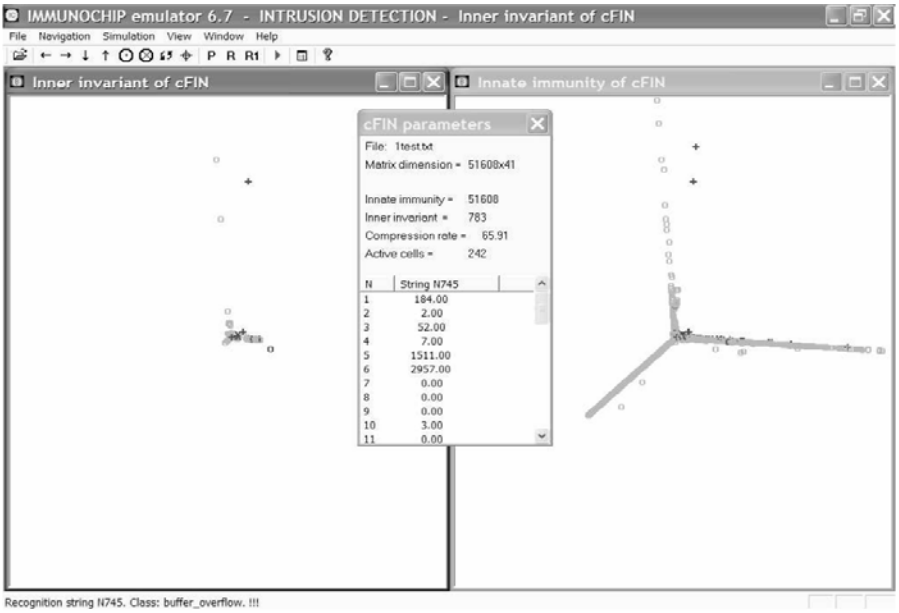


Fig. 12.2. Intrusion detection by cFIN: Antigen(String 745 of File 1.1) is mapped to 3D cFIN (bold skew cross in the centers of both screens) and recognized by the cytokine of the nearest cell of cFIN ("Class: buffer_overflow !!!" in the bottom status bar). Cells of cFIN related to the attacks are designated by bold "+"; normal class cells are designated by "o." Note that three clumps of "normal" cells of cFIN ("Innate immunity" in right-hand screen) look like "tumors" eliminated by apoptosis and autoimmunization ("inner invariant" in the left-hand screen).

File 1 is the training data file. It contains 51,608 network connection records. Any record (file string) has the following format, where parameters 2, 3, 4, 42 are symbolic, whereas the other 38 parameters are numerical (real values):

```
1) duration, 2) protocol_type, 3) service, 4) flag,
5) src_bytes, 6) dst_bytes, 7) land, 8) wrong_fragment,
9) urgent, 10) hot, 11) num_failed_logins, 12) logged_in,
13) num_compromised, 14) root_shell, 15) su_attempted,
16) num_root, 17) num_file_creations, 18) num_shells,
19) num_access_files, 20) num_outbound_cmds, 21) is_host_login,
22) is_guest_login, 23) count, 24) srv_count, 25) serror_rate,
26) srv_serror_rate, 27) rerror_rate, 28) srv_rerror_rate,
29) same_srv_rate, 30) diff_srv_rate, 31) srv_diff_host_rate,
32) dst_host_count, 33) dst_host_srv_count,
34) dst_host_same_srv_rate, 35) dst_host_diff_srv_rate,
36) dst_host_same_src_port_rate,
37) dst_host_srv_diff_host_rate, 38) dst_host_serror_rate,
39) dst_host_srv_serror_rate, 40) dst_host_rerror_rate,
41) dst_host_srv_rerror_rate, 42) attack_type.
```

For example, two records (# 1 and # 745) from File 1 are as follows:

```
0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,
0.00,0.00, normal.

184,tcp,telnet,SF,1511,2957,0,0,0,3,0,1,2,1,0,0,1,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,3,1.00,0.00,1.00,0.67,0.00,
0.00,0.00,0.00, buffer_overflow.
```

- File 1.1 was also prepared with the same 51,608 records in the same format but without the last parameter 42) attack_type.
- File 2 contains 31,1079 records in the same format as in File 1.1.
- File 1.1 and File 2 are the test data files.

Note that the KDD archive does not indicate the correct types of attack for any of the records of File 2. The only available information on possible attacks is gathered in Table 12.1 (column "Code" is the emulator's code of attack). Nevertheless, File 2 has been used to test whether the emulator is able to detect unknown intrusions, which had not been presented in the training data of File 1.

The results of training the emulator by File 1 are shown in Fig. 12.2, where the right-hand screen represents the initial population of cFIN after SVD (innate immunity of cFIN: $|W_0| = 51{,}608$), while the left-hand screen shows cFIN after apoptosis and

Table 12.1. Attack types.

Code	Attack type	File 1	File 2	Code	Attack type	File 1	File 2
0	normal	+	+				
1	apache2		+	16	pod	+	+
2	back	+		17	portsweep	+	+
3	buffer_overflow	+	+	18	rootkit	+	
4	ftp_write			19	saint		+
5	guess_passwd		+	20	satan	+	
6	imap			21	sendmail		+
7	ipsweep	+	+	22	smurf	+	
8	land	+		23	snmpgetattack		+
9	loadmodule			24	spy		
10	multihop		+	25	teardrop	+	
11	named		+	26	udpstorm		+
12	Neptune	+		27	warezclient		
13	nmap			28	warezmaster		
14	perl			29	xlock		+
15	phf	+	+	30	xsnoop		+

autoimmunization ("inner invariant of cFIN": $|W_1| = 783$). Total training time (for AMD 1.5 GHz) is 62 s, including 8 s for the first stage (SVD) and 54 s for the second stage (apoptosis and autoimmunization).

During the recognition of the records of File 1.1 and File 2, the emulator writes test results into the output file in the format: Record # - attack_type. For example, four records (## 744-747) with test results for File 1.1 are as follows (see also Table 12.2):

```
744 - normal.
745 - buffer_overflow. !!!
746 - buffer_overflow. !!!
747 - normal.
```

The emulator also shows on-line projection of any pattern to 3D space of cFIN (see bold skew cross on both screens) and writes the recognition result on the bottom panel (see "Class: back !!!").

Test results in Table 12.2 correspond completely to the correct attack types (parameter 42) of File 1.

Another test has been performed over File 2 to check whether the emulator is able to detect unknown intrusions, which had not been presented in the training data of File 1. The intrusion is treated as unknown if the projection of the corresponding pattern to cFIN lies outside the unit cube (according to Definition 1). The emulator recognized 13 unknown intrusions as the following records ## of File 2:

```
417, 12674, 97891, 139795, 170498, 176201, 177958, 232570,
236975, 296561, 296657, 96796, 297658.
```

According to Table 12.1, any unknown intrusion can correspond to one of the following types of attack that had not been presented in the training data:

```
apache2, guess_passwd, multihop, named, saint, sendmail,
snmpgetattack, udpstorm, xlock, xsnoop.
```

The recognition time per record is 15.7 ms for both tests of File 1.1 and File 2. This time covers not only computations but mainly reading the record from the test file, visualization of the recognition result (projection of the pattern to cFIN) on both screens of the emulator, and writing the result into the output file.

12.6 Forecast of Hydrophysical Fields in the Ocean

This section proposes a new method for identifying cellular automata (CA) using the IC approach to pattern recognition (described in Sec. 12.4). The essence of the method is the representation of states and transitions of CA by using FIN and the faultless reducing of the number of transitions by apoptosis and autoimmunization. The task is formulated using an analogy to the computation of the ecological atlas (Tarakanov and Tarakanov 2004), as well as the reconstruction of the hydrophysical field (Tarakanov et al. 2007). This approach is compared with the existing methods of neurocomputing in

Table 12.2. Test results for File 1.1.

Records ##	attack_type	Records ##	attack_type
745-746	Buffer_overflow	38036-38051	ipsweep
3095-7373	Smurf	38052-38151	back
9520-9523	Buffer_overflow	38302-38311	ipsweep
9590-9591	rootkit	42498-42519	ipsweep
9928-10007	neptune	42548-42567	ipsweep
10072	satan	42593-42594	ipsweep
10320	phf	42706-42708	ipsweep
13340-13519	portsweep	42730-42761	ipsweep
13569	land	42762-42770	buffer_overflow
13845-13864	pod	42771-42772	land
16326-16327	pod	42773-43385	neptune
17446-37902	neptune	44451-44470	neptune
37929-37939	ipsweep	44800-48452	smurf
37959-37963	ipsweep	48453-48552	teadrop
38005-38012	ipsweep	All other	normal

computational intelligence and the more conventional interpolation by the least-square method (LSM). The numerical example utilizes a real-world atlas of the sea surface temperature (SST) from (NOAA 1998).

According to Adamatzky (1994), identification of CA solves the problem of how to study the local behavior of cells from temporal slices of global evolution.

Let $c_{i,j,k} \in C$ be a set of CA cells with coordinates determined by indices i, j, k. Let $u(c) \subset C$ be a set of the nearest neighbors defined for any cell. Let c^t be the state of that cell at a discrete time step $t \in N$. Let us form a set of parameters (state vector) for any cell $X = [x_1 ... x_n]'$ that can include current and/or previous states of the cell $c^t,\ c^{t-1}, \ldots,\ c^{t-p}$ and/or states of its nearest neighbors: $u^t,\ u^{t-1}, \ldots,\ u^{t-p}$. Note that values of some parameters can be unknown for some cells and/or time steps.

Let values of the transition function $c^{t+1} = f(c^t,\ u^t)$ be given for some subset of cells $C_0 \subset C$ and time steps $N_0 \subset N$. The task is to determine the state of any cell at any time. Note that in real-world applications, the parameters $x_1, ..., x_n$ and the function f can be real-valued, whereas functional dependence $f(x_1, ..., x_n)$ can be too complicated or even unknown.

According to Section 12.4, we consider a special case of the general IC algorithm, which solves the task of identification of CA by given training vectors $X_1, ..., X_m$ and corresponding values of the transition function $f_1, \ldots, f_m$.

1. Form training matrix $A = [X_1 ... X_m]'$ of dimension $m \times n$.
2. Compute first q singular values and corresponding left and right singular vectors by SVD of the training matrix.

3. For any training vector X_i, compute its mapping $Y(X_i) = [y_{i1}...y_{iq}]'$ to FIN q-dimensional space.
4. Using apoptosis and autoimmunization, reduce m FIN training points to $k \leq m$ points.
5. Consider k points of FIN $Y_1, \ldots, Y_k$ together with their classes $f(Y_1), \ldots, f(Y_k)$ as the identified CA.
6. For any n-dimensional vector Z, compute its mapping $Y(Z) = [y_1...y_q]'$ to FIN q-dimensional space.
7. Among the reduced training points $Y_1, \ldots, Y_k$, determine the nearest one to $Y(Z)$:

$$Y_{i^*} = \min_{i=1}^{k} \|Y_i - Y(Z)\|.$$

8. Assign the class of the nearest point Y_{i^*} to the vector Z:

$$f(Z) = f(Y_{i^*}).$$

Steps 1–5 identify CA, whereas Steps 6–8 compute the value of the transition function $c^{t+1} = f(c^t, u^t)$ for any cell of CA at any time step. Note also that the existence, invariance, and faultlessness of such identification are guaranteed by the propositions and their proofs in Section 12.3.

Real-world data for the following numerical example have been obtained from the computer atlas of the Barents Sea (NOAA 1998).

Consider a 2D lattice $c_{ij}, \quad i = 1, \ldots 6, \ j = 1, \ldots, 16$, which is determined by the northern latitude $B_i = 75^\circ - (i-1)$ and the eastern longitude $L_j = 30^\circ + (j-1)$, where c^t is the monthly average SST in degrees centigrade ($\mathrm{T}^\circ\mathrm{C}$), which is given for all months: $t = 1, \ldots, 12$. Let us identify the CA (CA-SST) that will forecast the field of SST c^t_{ij} for any $t = n \mod 12$, $n \in N$. For example, a screen shot of the SST field for August ($n = 8, 20, 32, \ldots$) is shown in Fig. 12.3.

Let us define state vector $X = [x_1 \ldots x_n]'$ and transition function f for any cell so that $f = c^{t+1} - c^t$, $x_1 = c^t - c^{t-1}, \ldots, x_n = c^{t-(n-1)} - c^{t-n}$. Thus, the behavior of CA-SST can be completely described by $6 \times 16 \times 12$ training vectors $X_1, \ldots, X_{1152}$ and corresponding values of the transition function $f_1, \ldots, f_{1152}$. However, such CA may be nondeterministic since different values of the transition function $f_{i1} \neq f_{i2}$ may correspond to identical vectors $X_{i1} = X_{i2}$. The IC algorithm can identify such conflicting transitions of CA by using only the first-stage training Steps 1–3 (without apoptosis and autoimmunization) and testing of the identified CA by Steps 6–8. Then any mistake reveals the conflicting rules of CA. The number of such conflicts vs. the memory size of the CA-SST is shown in Table 12.3.

Table 12.3. Number of conflicts in CA-SST identified by FIN.

Memory size of CA (*n*)	2	3	4	5	6	7	8	9	10	11
Number of conflicts	761	569	405	323	238	160	91	61	33	0

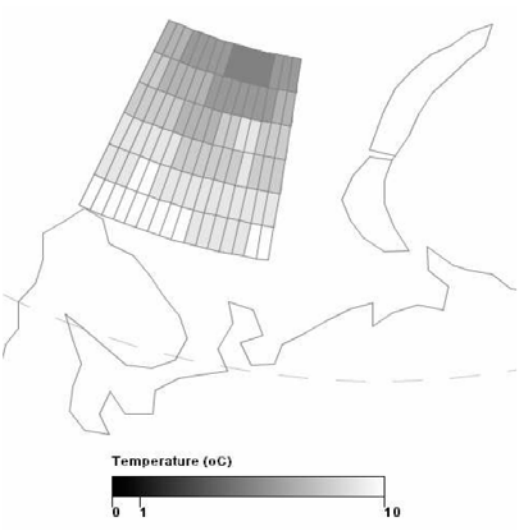


Fig. 12.3. A fragment of CA for the forecast of SST of the Barents Sea.

Thus, the deterministic CA-SST is possible only for $n \geq 11$ and the IC algorithm with apoptosis and autoimmunization (Steps 1–5) identifies that CA-SST by reducing 1152 transition rules in 11-dimensional space X to 646 points of 3D FIN ($q = 3$).

After the identification (of the minimal memory size) of the deterministic CA-SST, artificial neural network (ANN) and LSM can be utilized for the comparison with FIN.

ANN has been taken from Tarakanov and Tarakanov (2004). Its output f is computed by the following equations (so-called feedforward run):

$$Y = \sigma(W X_b), f = c_f \sigma(V' Y_b),$$

where σ is a function of the activation of neurons (so-called sigmoid):

$$\sigma(x) = \frac{2}{1 + \exp(-x)} - 1.$$

X_b and Y_b are vectors of input and hidden neurons with the bias $b = -1$:

$$X_b = \frac{1}{c_X} [x_1 \quad ... \quad x_n \quad -c_X]', \quad Y_b = [Y \quad -1]'.$$

$c_f = |f_{\max}|$ and $c_X = |x_{\max}|$ are scaling coefficients that provide compatibility with the diapason of the sigmoid; W is the weight matrix of input neurons; and V is the weight vector of hidden neurons. Both the weight matrix and the weight vector are trained by the error back propagation (EBP) method. The training cycle of ANN consists of m runs to consider all training vectors $X_1, ..., X_m$, whereas any run for training vector X_i consists of (a) the feedforward run and (b) the EBP correction of weights.

The error of the output neuron after the feedforward run is calculated by the following equation:

$$d_0 = (f_i^* - f_i)(1 - f_i^2),$$

where i is number of the training vector, f_i is value of the training function computed by ANN, and f_i^* is the given value of the training function. The error of the kth neuron in the hidden layer is calculated by the following equation:

$$d_k = d_0(1 - y_k^2)v_k,$$

where y_k is output of the neuron, v_k is the value of kth component of the weight vector V, $k = 1, ..., n_k$, and n_k is the number of hidden neurons. The weight vector and the weight matrix are corrected by the following (gradient) equations:

$$\Delta V = \eta d_0 Y_b, \quad \Delta w_{ji} = \eta d_k x_{ij},$$

where η is the so-called learning constant and x_{ij} is the value of the jth component of the input vector X_i. Training cycles are repeated until the total (training) error becomes less than some given value:

$$\frac{1}{m}\sum_{i=1}^{m}(f_i^* - f_i)^2 < e_0.$$

LSM calculates output $f = CX$ by the following vector of coefficients:

$$C = A^+ F,$$

where $F = [f_1 ... f_m]'$ is the vector of training values of function f, and A^+ is the so-called pseudoinverse of the training matrix:

$$A^+ = (A'A)^{-1}A'.$$

Comparative accuracy of three different methods (FIN, ANN, LSM) is presented in Table 12.4 based on the mean square error (MSE):

$$e = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(f_i - f_i^*)^2},$$

where $k = 1152$ for CA-SST.

For more representative examples, two other CA (so-called "chemical waves" and "solitons") have been identified by FIN using the data from Adamatzky (2001). A comparison of the accuracy of FIN with ANN and LSM is given in Table 12.4.
Note that neither ANN nor LSM is able to identify the conflicting states of CA.

Table 12.4. MSE of identification of CA.

Type of CA	FIN	ANN	LSM
Chemical waves	0.00	0.69	0.82
Solitons	0.00	0.26	0.38
SST of the Barents Sea	0.00	0.05	0.90

12.7 Discussion

According to the results that were obtained for intrusion detection (Sec. 12.5), FIN reduces the storing patterns by 65.9 times using apoptosis and autoimmunization without any loss of accuracy of recognition. Although this increases the training time (from 8 s to 1 min for AMD 1.5 GHz), it is, nevertheless, more important that the recognition time is decreased at least by 60 times per pattern by decreasing the number of the stored FIN cells that are to be compared with the recognition pattern.

It is worth noting that such a good performance of FIN (error-free recognition with rather short training time) on the data of real-life dimension looks unobtainable for the main competitors in the field of computational intelligence such as ANN and genetic algorithms (GA). According to our previous comparisons in (Tarakanov and Tarakanov 2004, 2005), FIN trains at least 40 times faster and recognizes correctly more than twice as often as ANN or GA in the tasks of environmental monitoring and laser physics. These applications demonstrate not only error-free recognition of the training set (as in Secs. 12.5 and 12.6), but mainly the advanced accuracy of FIN on the test set, which may differ significantly from the training one. These tasks have rather low dimensions: $17 \times 23 \times 6$ for the ecological atlas and 19×5 for the laser diode. The drawbacks of ANN and GA become especially unacceptable for the task of intrusion detection with the rather high dimension $51,608 \times 41$ and more.

It is also worth noting that FIN differs essentially from the negative selection algorithm (NSA) widely used in the field of AIS (Dasgupta 1999; de Castro and Timmis 2002). Actually, NSA aims to provide a set of detectors for self-nonself discrimination, whereas FIN guarantees a minimal set of cells for the correct recognition of any number of classes based on cytokines. Apparently, this makes FIN advantageous not only for intrusion detection on-line (Tarakanov et al. 2005b), but also for medically oriented applications to simulate cancer-specific apoptosis (Goncharova et al. 2005). Moreover, cytokines modulate proliferation and apoptosis of thymic cells as well as intrathymic T cell differentiation that includes not only negative but also positive selection (Savino and Dardenne 2000). Therefore, FIN also seems to be better suited for these kinds of simulations.

The results that were obtained (in Sec. 12.6) also show a clear advantage to the use of FIN for identification of CA over both neurocomputing and the more conventional method of interpolation. They confirm the advantages of an IC approach over other methods that were revealed by its application to the reconstruction of hydrophysical fields (Tarakanov et al. 2007). These advantages are expected to increase drastically with the increase in the dimension of training data.

We also point out that any run of IC with fixed dimension of FIN q and a fixed number of nearest FIN points for interpolation p gives the same MSE. Thus, FIN needs only $q \times p$ runs to determine the optimal parameters q^*, p^*, which provide the minimal error for any specific application, where $q \leq r$, $p \leq r$, while r is the rank of the training matrix. On the other hand, different training runs of ANN with a fixed number of hidden neurons n_k, learning constant η, and training error e_0 usually give a different MSE. This makes ANN somewhat unpredictable and dictates its statistical characterization.

Moreover, ANN is too slow in comparison with FIN and conventional interpolation. For example (Tarakanov et al. 2007), IC needs just 21 runs (about 20 s) to obtain optimal parameters of FIN ($q^* = 3, p^* = 5$), whereas ANN needs 1750 runs (about 24 h!) for the same purpose ($n_k^* = 3$, $\eta^* = 0.01$), without any guarantee that, say, 20 or 100 hidden neurons may not minimize total error.

Table 12.4 shows that FIN is more accurate than ANN and LSM. Similar results in Tarakanov et al. (2007) also demonstrate the theoretically rigorous feature of training FIN with zero error rate, whereas training errors of ANN and LSM are usually too high. In addition, attempts to reduce training errors of ANN may lead to the so-called overtraining effect when total error increases drastically.

The memory constraints of FIN and ANN look more comparable. For FIN, they are determined mainly by the dimensions of the training matrix $(m \times n)$ and FIN $(m \times q)$, i.e., by the number and the dimension of training vectors and the dimension of FIN space. The memory constraints of ANN are not much lower and are determined mainly by the dimension of the weight matrix $(n_k \times n)$, i.e., by the number of hidden neurons and also by the dimension of the training vectors. However, there is no need to store the training matrix after FIN has been trained. So, the memory constraints of IC and ANN can be compared by the dimensions of FIN and the weight matrix, respectively.

In conclusion, the results that were obtained confirm similar advantages of FIN over other methods of computational intelligence and more conventional methods of interpolation revealed by their applications to real-world data of information assurance, ecology, laser physics, and hydroacoustics. Possible ways to reinforce these advantages may be norms other than Euclidian together with more delicate methods of interpolation by nearest points of FIN. The advantages of the proposed approach coupled with its advantages for 3D modeling (Tarakanov and Adamatzky 2002) make FIN rather promising for on-line simulation of real-world 3D fields.

References

Adamatzky, A. (1994). *Identification of Cellular Automata*. Taylor & Francis, London.

Adamatzky, A. (2001). *Computing in Nonlinear Media and Automata Collectives*. IOP, Bristol and Philadelphia.

Ader, R., Felten, D.L., and Cohen, N., editors (2001). *Psychoneuroimmunology*. Academic Press, New York.

Agnati, L.F., Tarakanov, A.O., Ferre, S., Fuxe, K., and Guidolin, D. (2005a). Receptor-receptor interactions, receptor mosaics, and basic principles of molecular network organization: Possible implication for drug development. *Journal of Molecular Neuroscience*, 26(23):193–208.

Agnati, L.F., Tarakanov, A.O., and Guidolin, D. (2005b). A simple mathematical model of cooperativity in receptor mosaics based on the "symmetry rule." *Biosystems*, 80(2):165–173.

Balkwill, F., editor (2000). *The Cytokine Network*. Oxford University Press, New York.

Bay, S.D. (1999). *The UCI KDD Archive*. University of California, Department of Information and Computer Science, Irvine, CA. Available at http://kdd.ics.uci.edu.

Bunk, S. (2003). Signal blues: stress and cytokine levels underpin a provocative theory of depression. *The Scientists*, 25:24–28.

Cloete, I., and Zurada, J.M., editor (2000). *Knowledge-Based Neurocomputing*. MIT Press, Cambridge, MA.

de Boer, R.J., Segel, L.A., and Perelson, A.S. (1992). Pattern formation in one- and two-dimensional shape space models of the immune system. *Journal of Theoretical Biology*, 155:295–333.

de Castro, L.N., and Timmis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, London.

Dasgupta, D., editor (1999). *Artificial Immune Systems and Their Applications*. Springer, Berlin.

Fisher, P.B., et al. (2003). mda-7/IL-24, a novel cancer selective apoptosis inducing cytokine gene: From the laboratory into the clinic. *Cancer Biology and Therapy*, 2:S023–S037.

Goncharova L.B., Jacques Y., Martin-Vide C., Tarakanov A.O., and Timmis J.I. (2005). Biomolecular immune-computer: theoretical basis and experimental simulator. *Lecture Notes in Computer Science,* 3627:72–85. Springer, Berlin.

Horn, R., and Johnson, Ch. (1986). *Matrix Analysis*. Cambridge University Press, Cambridge England.

Igney, F. H., and Krammer, P. H. (2002). Immune escape of tumors: Apoptosis resistance and tumor counterattack. *Journal of Leukocyte Biology*, 71(6):907–920.

Jerne, N.K. (1973). The immune system. *Scientific American*, 229(1):52–60.

Jerne, N.K. (1974). Toward a network theory of the immune system. *Annals of Immunology*, 125C:373–389.

Kourilsky, P., and Truffa-Bachi, P. (2001). Cytokine fields and the polarization of the immune response. *Trends in Immunology*, 22:502–509.

Kurzrock, R. (2001). Cytokine deregulation in cancer. *Biomedecine & Pharmacotherapy*, 55(9/10):543–547.

NASA (2004). Human immune system inspires NASA machine-software fault detector. *NASA Bulletin, October 26*.

NOAA-NESDIS-National Oceanographic Data Center (1998). *Climatic Atlas of the Barents Sea*. Available at http://www.nodc.noaa.gov/OC5/barsea/barindex.html.

Savino, W., and Dardenne, M. (2000). Neuroendocrine control of thymus physiology. *Endocrine Reviews*, 21(4):412–443.

Tarakanov, A., and Adamatzky, A. (2002). Virtual clothing in hybrid cellular automata. *Kybernetes*, 31(7/8):394–405.

Tarakanov, A., and Dasgupta, D. (2000). A formal model of an artificial immune system. *BioSystems*, 55(1–3):151–158.

Tarakanov, A., and Dasgupta, D. (2002). An immunochip architecture and its emulation. *NASA/DoD Conference on Evolvable Hardware (EH'02)*, 261–265. IEEE Computer Society, Los Alamitos, CA, USA.

Tarakanov, A.O., Goncharova, L.B., and Tarakanov, O.A. (2005a). A cytokine formal immune network. *Lecture Notes in Artificial Intelligence*, 3630:510–519. Springer, Berlin.

Tarakanov, A.O., Kvachev, S.V., and Sukhorukov, A.V. (2005b). A formal immune network and its implementation for on-line intrusion detection. *Lecture Notes in Computer Science*, 3685:394–405. Springer, Berlin.

Tarakanov, A., Prokaev, A. and Varnavskikh, E. (2007). Immunocomputing of hydroacoustic fields. *International Journal of Unconventional Computing* 3(2): 123–133.

Tarakanov, A.O., Skormin, V.A., and Sokolova, S.P. (2003). Immunocomputing: *Principles and Applications*. Springer, New York.

Tarakanov, A.O., and Tarakanov, Y.A. (2004). A comparison of immune and neural computing for two real-life tasks of pattern recognition. *Lecture Notes in Computer Science*, 3239: 236–249. Springer, Berlin.

Tarakanov, A.O., and Tarakanov, Y.A. (2005). A comparison of immune and genetic algorithms for two real-life tasks of pattern recognition. *International Journal of Unconventional Computing*, 1(4):357–374.

Vilcek, J., and Feldman, M. (2004). Historical review: Cytokines as therapeutics and targets of therapeutics. *Trends Pharmacological Science*, 25:201–209.

Wall, L., Burke, F., Caroline, B., Smyth, J., and Balkwill, F. (2003). IFN-gamma induces apoptosis in ovarian cancer cells in vivo and in vitro. *Clinical Cancer Research*, 9:2487–2496.

13

A Model for Self-Organizing Data Visualization Using Decentralized Multiagent Systems

Andrew Vande Moere

13.1 Introduction

In the information society of today, corporations, government agencies, and various scientific fields are continuously accumulating data. The complexity of these data is staggering, and our ability to collect it is increasing faster than our ability to analyze it. Although current database technology has made it possible to store and manage huge amounts of data in a comprehensive manner, the exploration of these data is still bound to relatively rigid interfacing methods. *Visualization*, the representation of data graphically rather than textually, aims to exploit the high-bandwidth human perceptual and cognitive capabilities to draw interferences from visual form. Its real purpose goes beyond that of generating beautiful pictures, as visualization aims to induce useful insights where there were none before. Such insights can take different forms, such as through the discovery of unforeseen data phenomena, the ability to derive decisions or to communicate knowledge visually, and insights based on discoveries made within the dataset. More particularly, the field of *data visualization* faces the need to represent the structure of and the relationships within large, complex datasets that contain so-called "abstract" concepts, which lack a natural notion of position in space (Card et al. 1999; Chi 2000; Tory and Möller 2004).

Thus data visualization differs from *scientific visualization*, which generally represents datasets that have a physical form in nature and generally can be represented through a process of graphical reproduction, such as geographical layouts, wind simulations, or medical imaging. As abstract data are nonspatial and lack any natural representation, the fundamental challenge for data visualization is thus "... how to invent new visual metaphors for presenting information and developing ways to manipulate these metaphors to make sense of the information" (Eick 2001). Data visualization is different than *data mining*, which deals with the analysis of datasets to establish relationships and identify patterns, so that data items are clustered into groups according to apparent data similarity. Although data visualization predominantly aims to convey meaning and insights, it often uses data mining techniques to analyze or order the dataset first. As datasets continuously become more complex in terms of size, time variance, and data dimensionality, data visualization techniques

need to evolve to accommodate the more sophisticated methods of data analysis and visual representation.

Self-organization is a process in which the internal structure of a complex system automatically increases without being guided by an outside source. Self-organizing processes are often coupled to the occurrence of *emergent* phenomena. A property is generally characterized as emergent when perceived complex behavior arises out of a collective of entities that individually were not 'explicitly' programmed to do so. Emergent behavior is usually generated by the continuous and recursive interaction of individual entities with similar entities in their immediate environments. When well-considered interaction mechanisms are used, these interactions can, on a holistic level, lead to seemingly rational behavior and the observable proliferation of order. Probably the best-known examples of such emergent behaviors are the flocking of birds and the schooling of fish, or the amazing organizational capabilities (e.g., shortest way finding, waste disposal) of ant colonies. Principles of emergence generation have been successfully applied to a wide range of fields, including problem solving, learning, and optimization problems in the areas of system design, pattern recognition, biological system modeling, signal processing, decision making, simulation, and identification (Kennedy and Eberhart 2001). Such systems are able to successfully solve complex problems that contain multiple unpredictable or time-varying parameters. Self-organization intelligence is basically divided and distributed to simple and comprehensible units, which holistically are tolerant of local failures or changing environments. Self-organization thus seems to be an ideal method for use in data visualization, as this field often involves multifaceted and inherently unpredictable datasets that need to be 'ordered' into apparently well-ordered diagrams. By applying self-organization to the problem of data visualization, data mapping metaphors might emerge that suit particular dataset characteristics, thereby potentially revealing data patterns that were previously unknown.

Self-organizing data visualization is based on augmenting direct data mapping techniques with simple forms of artificial intelligence, assuming that datasets inherently contain enough characteristics to organize themselves. It aims to instigate visual emergent phenomena from meaningful data relationships inherent in the dataset. It is based on the concept that each single data item within a dataset can be mapped onto a unique *data visualization agent*. Each agent then determines its own visual properties through a process of continuous and recursive interactions with other agents. The iterative organization of the agents creates emergent visual effects that are based on data properties that are autonomously detected by agents and can be perceived and interpreted by users. Ultimately, this approach might lead to new data visualization techniques that are more 'aware' of the data characteristics they represent, and can optimize the overall representation according to good visualization guidelines and human visual cognitive knowledge.

This chapter demonstrates the potential of self-organization for data visualization purposes. At the same time, it also illustrates how self-organization probably is not the most efficient existing method to represent data, visually because of the required computation power and the considerable amount of effort needed to manage the various parameters that control the emergent phenomena. As is discussed in Section 13.5,

self-organizing visualization suffers from inadequate calculation performance, comprehensibility difficulties, and the influence of multiple parameters on the emergent effects. However, self-organizing data visualization is nonetheless a valuable alternative approach to the predetermined and fixed data mapping techniques that currently exist, as it specifically allows for the occurrence of unexpected visual organizations.

The principles of the self-organizing data visualization model are described and illustrated with three different case studies. The 'information particle,' 'information flocking', and 'cellular ant' approaches use different visual metaphors, based, respectively, on particle animation, swarming, and cellular automata principles. Each system demonstrates how meaningful properties contained within datasets can be made apparent by the simulation of visually emergent effects using relatively simple self-organizing behavior rules. Based on these findings, Section 13.5 discusses the potential benefits and shortcomings of this concept.

13.2 Background

Self-organizing data visualization is fundamentally different from most other agent-based approaches known in the fields of visualization and data mining, which tend to focus on using agent intelligence for data analysis, visualization data flow, or software development optimization.

13.2.1 Decentralized Multiagent Systems

An *agent* is a system situated within an environment. It senses that environment and acts on it autonomously, over time, to realize a set of design objectives (Franklin and Graesser 1996). An *intelligent agent* typically can be characterized as social, reactive, and proactive, as it operates in a changing, unpredictable environment where there is a possibility that actions can fail (Woolridge 2001). Such agents can collaborate with other agents, can perceive and respond to changes, and can exhibit goal-directed behavior. A *multiagent system* is populated with multiple equal agents, generally because they pursue different goals or because the environment is too complex for a single agent to observe it completely. The agents within such system interact and negotiate with each other, a process that is generally determined by concepts of cooperation, coordination, and negotiation. An agent's behavior can be determined by a *rule-based system* that interprets communications and interactions with other agents, the perceived environmental changes, and the agent's goals. A *decentralized multiagent system* contains numerous equal agents that have communication links with those in their neighborhood, either directly or through the environment, but always in the absence of a centralized coordinator. Such systems are designed to facilitate self-organization, the spontaneous increase in complexity of internally organized structures.

Accordingly, self-organizing data visualization is based on decentralized multiagent systems that are inherently capable of simulating collective behavior, such as swarming, cellular automata, and particle animations. By controlling these agents with data values, the resulting behaviors are fully controlled by dataset properties. The

resulting visual 'effects' are presented in an organized and consistent manner, so they can be perceived and interpreted efficiently.

13.2.2 Data Visualization

Data visualization is based on the principle that meaningful data properties, such as tendencies, trends, outliers, and similarities among data items, can be made apparent by translating raw textual or numerical data values into a holistically interpretable visual representation. As shown in Fig. 13.1, the typical data flow process for translating data into visual form is considered to be an iterative analysis cycle that passes through four distinct phases (Upson et al. 1989). First, a large amount of 'raw' data is filtered into manageable and interpretable data subsets in the Data Space domain. The filtering of these subsets is determined by dataset quality and user interests. First, corrupt or otherwise invalid data items, which might have resulted from faulty parsing or querying procedures, are removed from the dataset. Then, the system caches only those data items that are relevant for a particular user, for instance, the particular data items that are visible or are selected by settings in the user interface. The derived data subsets in Feature Space are then translated into visual distinguishable forms in Object Space, as specifically designed data mapping rules generate visual artifacts that are consistent with the values and attributes within the dataset. In general, specific data attributes relate to visual objects (e.g., points, lines, shapes), whereas their values determine the transformations applied to those forms (e.g., position, color, direction, size). As each visual object stands for a unique data item, the resulting visual constellation of objects is representative for the whole dataset. Finally, the resulting collection of visual objects, labels, and legends has to be rendered into a coherent perceivable form in Image Space, as different forms of media require specific treatments and formats, depending on aspects such as the display size, distributed computation resources, user context, or hardware capabilities. As is described in the next section, agent-based systems have been successfully used in each of these visualization data flow stages.

13.2.3 Agent-Based Visualization Approaches

Applied artificial intelligence and the visualization domain have been successfully combined for various purposes. However, most past work has focused on either the filtering (Feature Space) or rendering (Image Space) phases of the typical visualization

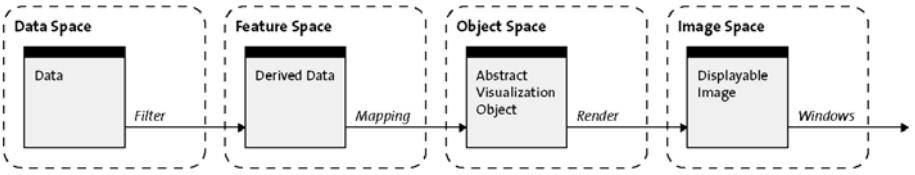


Fig. 13.1. Typical information visualization data flow model (after (Upson et al. 1989)).

data flow. In contrast, the proposed model in this chapter uses agents for the mapping of derived data to visualization objects.

Agent-based data mining addresses the retrieval, management, and extraction of information out of large, distributed databases. Data mining agents are capable of accessing distributed datasets from which useful, higher-level information is extracted. Such agents keep track of and organize information spaces by understanding meaningful relationships within datasets, and are able to present these proactively to users. Some agent-based data mining algorithms are based upon a centralized system component, such as middle agents or directory services, which know the location and capabilities of each agent, enabling each agent to communicate with all other agents. Complex problems, such as clustering, are optimized by reducing the solution space that the algorithm has to search or by partitioning them into a series of smaller problems. In contrast, a decentralized multiagent system contains numerous equal agents that have some communication links to other agents. The goal of such agents is to iteratively select and rearrange these links to form interactive connections, so that data clustering can be achieved in a collective effort (Ogston et al. 2003).

Most so-called *agent-based visualizations* consist of traditional representation techniques that utilize the agents as their dataset to be visualized. These agents are thus not designed to interfere with the visualization, and the resulting representations convey typical agent properties instead of characteristics of complex, abstract datasets. For instance, so-called *multiagent visualizations* have been used to display intrinsic relations (e.g., number of messages, shared interests) among agents for monitoring and engineering purposes (Schroeder and Noy 2001). Visualization in data mining tends to be used to present final results, rather than playing an important part throughout the entire data exploration process (Robinson and Shapcott 2002). Multiagent systems have been developed to organize the visualization data flow, for instance, for representing complex fuzzy systems (Pham and Brown 2003).

Some visualization systems utilize agents to determine the most efficient display parameters within the Image Space. For instance, an advanced graphic visualization rendering pipeline can consist of several remotely dispersed agents to allow for the abstraction and reuse of rendering strategies, including distributed or progressive rendering (Roard and Jones 2006). Similarly, the e-Viz system is based on agents, so-called active visualization components, that are designed to self-heal software failures or network problems, and can self-optimize the rendering performance (Brodlie et al. 2006). Multiagent systems are then used to support flexibility regarding the interaction possibilities or the rendering quality of the visualization (Ebert et al. 2001). Agents and visualizations have been combined to organize, analyze, or mine abstract datasets, with agents embedded in both Data and Feature Space. Data can be ordered and filtered by so-called "visualization agents," which are then represented using conventional techniques, such as for discovering the most desired pages from a large website (Hiraishi et al. 2001). Agents can help users to decide on the most suitable visualization technique depending on the dataset characteristics (Marefat et al. 1997), guide users toward the most effective visualization approach according to dataset characteristics and visual cognitive guidelines (Senay and Ignatius 1994), or act as visualization assistants to help choose methods for effectively visualizing e-commerce data (Healey et al. 2001).

Similarly, some visualization approaches help developers choose the optimal association rules that allow for efficient data exploration (Sadok and Engelbert 2004).

Only a few approaches exist that use multiagent systems as a way to map data into visual form or, in other words, that contain agents in Object Space that act as visual elements themselves. In 1996, Ishizaki proposed the idea of deriving design solutions by the emergent behavior of a collection of agents, so-called *performers*, which are individually responsible for presenting a particular segment of information (Ishizaki 1996). Although this approach focused primarily on interactive and information-rich interfaces consisting of agents that represent news headlines, there are several conceptual similarities with our proposed data visualization model: the agents represent data by themselves, are not bound to particular types of visual expression, are governed by a set of behavior rules that are based on the detection of data characteristics, and even collaborate to reach a common strategic goal. More recently, a similar collaborative multiagent system was developed that produces artistic Mondriaan-like paintings emergently (Mason et al. 2004). Other research approaches that are more specifically relevant to each of the case study approaches are mentioned in their respective sections.

13.3 Emergence in Data Visualization

The self-organizing data visualization model is based on the assumption that data items themselves can be treated as agents that, depending on their inherent relationships, are capable of autonomously determining their own visual representation. Accordingly, this section investigates the specific requirements to derive emergent visual effects out of data characteristics.

13.3.1 Visual Emergence vs. Data Pattern Emergence

Each data visualization technique is uniquely determined by its *data mapping rules*, a set of simple conditions that 'translate' data values into visual form from Feature Space to Object Space. Typically, each single data item within a dataset is mapped onto a separate *visual object*, such as a point, line, shape, or three-dimensional object. The data variables of a data item then determine the *visual transformations* of this visual object, such as its position, color, shape, length, or orientation. Generally, these data mapping rules are implemented by the developer, who, at the very least, takes following aspects into consideration:

1. *Anticipated dataset characteristics*, in order to specify the visual objects and their properties, and design the according data mapping rules. Several empirically derived guidelines exist that correlate effective visualization techniques according to dataset typology (Mackinlay 1986) (e.g., numerical, categorical). However, the data mapping choice is typically made by its developer. Even when a specific technique has been chosen, a visualization developer has to fine-tune the data mapping rules according to the expected size, time-dependency, data dimensionality, time variance, data value range, and data pattern semantics of the dataset. For instance,

anticipated maximum and minimum data values have to correspond to the specific axes scales or color scale, while the dataset update frequency determines to how fast the visualization should adapt to any dynamic changes.

2. *User interests*: Similar to the dataset characteristics, the data mapping rules should assign the most dominant and preattentive visual properties, such as color and position, to the data patterns a user is most interested in.
3. Aspects of human *visual perception and cognition*, so that data patterns are represented in an easily perceived and intuitively understandable way. For a data visualization to be effective, it has to translate data patterns into artifacts that visually stand out, while allowing the user to comprehend their meaning. In other words, the data mapping has to allow for an 'inverse' mapping to occur rapidly and faultlessly by its users. Therefore, the data mapping design process needs to incorporate insights from different adjacent disciplines, such as perception in visualization (Ware 2000), visualization guidelines (Card et al. 1999), user interaction (Shneiderman 1998), data exploration (Jankun-Kelly et al. 2002), task-related and human factors (Tory and Möller 2004), and graphic design principles (Tufte 2001). As shown by insights from the Gestalt school of psychology, humans attempt to perceive and understand any graphical representation as one, single, coherent form or Gestalt, instead of a collection of individual components. The Gestalt research was an attempt to describe the principles of perceptual visual processes that result in perceptual coherence, which was synthesized in a set of so-called Gestalt laws. Figure 13.2 illustrates how groups of nearby objects, or objects similar in color or shape, are perceived as belonging together. Conceptually, this principle implies a level of *emergence* occurring on a perceptual level, as visually complex patterns can be recognized from a collective of entities that individually were not explicitly informed to do so.

The traditional design of data mapping rules is specifically motivated by the wish to highlight meaningful data phenomena by purposively simulating the occurrence of "visually emergent" effects. Traditional visualization approaches rely on the direct translation of similar data values to similar visual properties, so that, for instance, data items that are 'similar' tend to be represented near each other or are highlighted by an identical color. In contrast, a self-organizing system aims to achieve a visually similar effect by a decentralized approach, so that similar data items should first 'find' each other, and then 'stay close' together, or 'determine' a 'common' color. The detection of

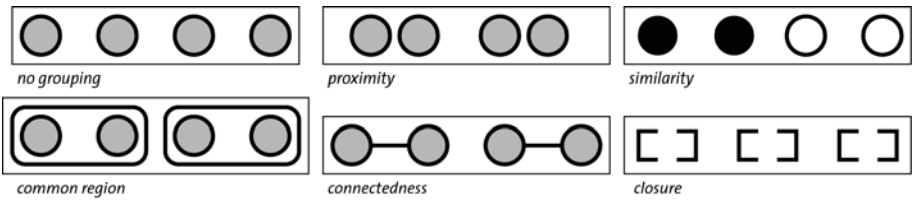


Fig. 13.2. Visual emergence as Gestalt rules. Separate objects are perceived as belonging together due to specific common characteristics.

similar items, their grouping, or their cooperative color choice can be considered emergent, as self-organization requires simple pairwise comparisons between data items instead of a centralized data analysis. Consequently, self-organizing data mapping is a process designed to magnify dataset emergent phenomena into visual emergent effects. This dual reliance on emergence, first in the detection of data patterns and then in the deliberate generation of visual forms, is the main driving force behind the proposed self-organizing visualization model.

Technically, self-organizing data visualization is based on mapping meaningful data values with the numerical parameters that influence emergent behaviors of applied artificial intelligence simulations. As those weighting values are directly derived from the dataset, the resulting emergent behavior represents that dataset. In contrast, traditional data visualization approaches are determined by rigid data mapping rules, predefined by the application developer. Any alterations to such data mapping rules tend to be strictly limited to the configuration of the visual object transformations, such as the application's color palette or the axes scales. Although most data visualization applications offer a set of interactive features facilitating typical "human-computer interaction mantra" operations (i.e., overview, zoom, and select) (Shneiderman, 1998), data mapping rules are generally considered to be a fixed part inherent to the visualization technique. In contrast, by merging the concept of emergence with data mapping, unforeseen behaviors that convey useful data patterns in unexpected ways might become apparent.

13.3.2 Data Visualization Agent

The self-organizing data visualization concept is based on mapping data items directly onto agents. Each single data item (e.g. *data object, data tuple*, or *database row,* retrieved from a database or dataset) is mapped onto a unique data visualization agent. The aim of this particular approach is to let agents "behave" according to their individual data values according or to any differences with the data values of their neighbors. As illustrated in Fig. 13.3, each agent is visually represented by a visual object, and its dynamic behavior (e.g., position, color, shape, speed) is determined by a set of behavior rules that in turn is controlled by its data item's data values. As time progresses, its data values change and the agent's dynamic behavior adapts accordingly.

Traditional data visualization techniques are generally based on preanalyzed datasets. For instance, some approaches are based on *data similarity tables*, which

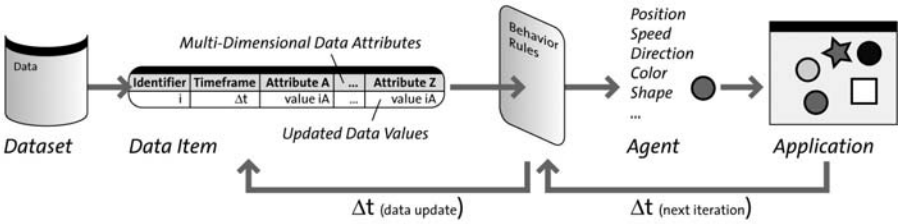


Fig. 13.3. A data visualization agent and its behavior as determined by data-driven behavior rules.

contain quantitative measurements of how pairwise data items relate to each other. In contrast, self-organizing data visualization performs the dataset analysis during the visualization itself, where it is executed by a collection of agents instead of by a single, central process. All agents exist in a shared visualization space, which is ruled by a common *application timeline*. All agents are 'situated,' viewing the world from their own perspective rather than from a global one. Their actions are determined by internal states, as determined by a set of common behavior rules. These rules represent the strategy of the agent and how it should behave according to its own data characteristics, those of neighboring agents, and to any external influences, such as real-time changes in the dataset or user interactions (e.g., selection, filters).

For *static* datasets, each agent continuously represents the same set of data values. For *time-varying* (also called *dynamic, time-dependent, time-variant, time-based*, or *temporal*) datasets, all agents are synced to the sequential progress of the application timeline, which moves 'forward' or 'backward' in an iterative way. Each agent is then subject to a data update, which corresponds either to a data alteration or to no change at all. The application timeline progresses according to a specific rhythm, so that newly updated data items are fetched from the database and then assigned to the corresponding agent. An updated agent will reconsider its visual transformations and might change its dynamic behavior or representation cues accordingly. Owing to the presence of multiple, equal agents that are not centrally controlled, the data visualization is essentially a decentralized multiagent system. In short, a typical data visualization agent needs to have at least following characteristics:

- *Data Interpretation*. Each agent is aware of all the data attributes and values of the data item it represents. It can also detect any changes that occur to it over time, caused by the application timeline simulation or by user interaction.
- *Local Perception.* The agent is capable of perceiving the environment it is present in, including any other agents in its vicinity and any external objects or space boundaries.
- *Local Communication.* Each agent can communicate with other agents nearby and can compare all the attributes of those agents, including their data item or their actual visual state.
- *Negotiation.* More 'intelligent' agents can perform complex negotiations with neighboring agents, such as simple tit-for-tat strategies (e.g., the shape of agent A grows while taking away the space required from the shape of its neighboring agent B) or positional swapping (e.g., agent A and B swap their positions).
- *Visual Presence Autonomy.* Each agent is capable of autonomously determining its own visual presence, in the form of altering the visual attributes it inherently possesses, including its spatial position, size, color, orientation, speed, direction, shape, and so on. According to the negotiation characteristic, agents can also change the visual properties of their neighboring agents to some degree.
- *Historical Memory.* Each agent can store historical events and access them, such as the previous values that its data item has contained, the coordinates it has passed through, the visual attributes it has adapted to, and the other agents it has encountered over time.

More complex agent characteristics that would be useful for data visualization purposes can be easily imagined, such as learning, reasoning, motivation, curiosity, and so on. However, this chapter focuses on the self-organizing and emergent capacities that can be achieved by using reasonably simple agent principles.

13.3.3 Behavior Rules

At initialization, all agents are positioned randomly on the visualization canvas, and they are all governed by the same set of behavior rules. These rules indirectly 'map' the agent's data values onto visual properties. For instance, one or more behavior rules could define an agent's color according to the relative difference in data values with one of its neighbors or let it move toward the most similar agent. As mentioned in Section 13.3.1, these behavior rules need to be specifically designed to simulate visually emergent effects, so that similar data items can be effectively perceived as 'belonging together'. For instance, the Gestalt rule of proximity states that objects that are located nearby each other are emergently perceived as one. In traditional data visualization, this particular characteristic is exploited by specifically choosing the data attributes so that data items with similar values are positioned in the vicinity of each other. For the agent-based approach, a similar visual effect is achieved by a well-considered sequence of individual agent actions. Owing to the multitude of such pairwise agent interactions, multiple similar agents will group and large spatial clusters will form. Although the end result of these different methods might seem similar at first sight, self-organizing data visualization is intrinsically dynamic, completely decentralized, and determined in real time.

13.4 Case Studies

The following section describes three different approaches of the self-organizing data visualization model. From a simple set of cause-and-effect behavior rules to more elaborate and complex interagent negotiation strategies, the proposed techniques demonstrate how the mapping of data onto visual properties can be accomplished without the need for central control. The simulation of self-organizing behavior in each technique is based on insights borrowed from the fields of applied artificial intelligence and artificial life. The resulting emergent behavior can be interpreted as meaningful data trends and patterns, as the apparent effects are fully determined by, and thus reflect, inherent dataset characteristics.

13.4.1 Metaphor 1: Infoticle Method

The *infoticle* system is based on a simple set of sequential behavior rules that determine the speed and direction of moving particles in a 3D virtual space. This particular approach demonstrates how the traditional vocabulary used by data visualization, consisting of position, color, shape, size, etc., can be effectively broadened with a novel

visual property known as *dynamic animation*. The power of animation to create interpretatively rich behaviors allows for the generation of distinctive motion typologies that convey time-varying data trends.

Self-Organizing Method: Particle Animation

A *particle* can be imagined as a pointlike mathematical object in 3D space. It is generally determined by a fixed set of attributes, such as position, velocity (speed and direction), acceleration, mass, color, lifespan, age, shape, size, and transparency. A *particle system* consists of a collection of particles, each of which is influenced independently through time by a set of predefined conditions (Martin 1999). Particle systems were first formally proposed by Reeves (1983) as a rendering technique that is able to realistically simulate dynamic natural phenomena such as rain, explosion, waterfalls, fire, or smoke. Currently, particle systems constitute a widely used computer graphics technique, as the essential programming logic required to efficiently implement real-time particle systems have been described in detail. [See Lander (1998); van der Burg (2000) for applicable software programming approaches.] Particle systems can be combined with so-called "*forces*," which are abstract, pointlike elements in virtual space that influence the movement and speed of each single particle by attracting or repulsing it according to the laws of Newtonian mechanics. Given correct initial conditions, and combined with internal relationships or external forces, particle systems can be animated over time to convey seemingly complex and intriguing behaviors (Tonnesen 2001). For instance, the combination of particles and forces enables computer graphic designers to convey realistic visual effects, from simulating gravity in space galaxies over the bouncing of balls on surfaces to the fading of flames.

The ultimate goal of using dynamic animation as an independent visual property in data visualization is the creation of interpretatively rich behaviors that seem to be intentional, possibly even provoking causality, animacy, and initiative. *Behavioral animation* techniques typically employ rule-based systems to specify the dynamic motions, so that a set of cause-and-effect rules is listed, which the elements that are being animated must follow. For instance, Lethbridge and Ware (1990) used simple behavior functions based on distance, velocity, and direction to model complex causal relationships. Empirical cognitive research suggests a huge potential for using motion for data visualization purposes (Ware et al. 1999). Bartram and Ware (2002) proved that motion typology has a strong perceptual grouping effect that can be effectively used for information display. For instance, similar motion typologies are perceived as grouped over time, a phenomenon also called *temporal grouping*, which relates to the appearance and locations of the elements, their proximity in time, and the similarity of motions (Kramer and Yantis, 1997). The use of motion for representing information might seem to be in contradiction with traditional mapping techniques that position data items on 'fixed' Euclidian coordinates. Rather, motion should be considered as a property that is independent of position, relying instead on the perceived dynamic relationships among the various elements that move similarly or differently. Instead of 'morphing' visual elements from one fixed Euclidian position to another, behaviorally moving particles have the inherent capability to be updated on-the-fly.

Particle Animation as Data Mapping

In the infoticle system, each data visualization agent is represented as a unique particle within a 3D virtual space. This particle is coined "infoticle" (short for information particle), and will be further referred to as "agent." The virtual world also contains a set of forces that are fixed points in space and represent specific static data attribute values that an agent's data item potentially can contain. The dynamic behavior of each agent is determined by a set of behavior rules, which in turn are triggered by the alterations of its data values over time. In the infoticle system, these behavior rules only specify: (1) to which particular force the agent is 'attracted', and (2) whether it should speed up or slow down. In practice, the application simulates the linear progression of a timeline that corresponds to the time stored in the datasets. As the timeline progresses, data objects are updated with new data values. As a result, some data items, and thus agents, are updated and some are not. For those that are updated, their data items may contain either different data values or equal ones. The infoticle behavior rules specify the following:

- *Data Update and Alteration*. Each agent whose data item is updated over time speeds up and is 'attracted' to a force that represents the new updated data value (see Fig. 13.4). Once it reaches within a particular distance of the force, it starts orbiting around it. This rule causes agents with similar data items to congregate in a circular motion around the same data attribute (force), similar to space satellites that orbit planets.
- *Data Update Only.* Agents that are updated but encounter no change in data value only speed up while being attracted to the same force. This rule causes these agents to follow a more elliptic path around the same data attribute (force), similar to space comets around our sun.
- *No Data Update.* Agents that are not updated are subjected to a virtual 'friction' and slow down, while being attracted to the same force. Such agents tend to 'fall back' and eventually 'crash' into the center of the force.

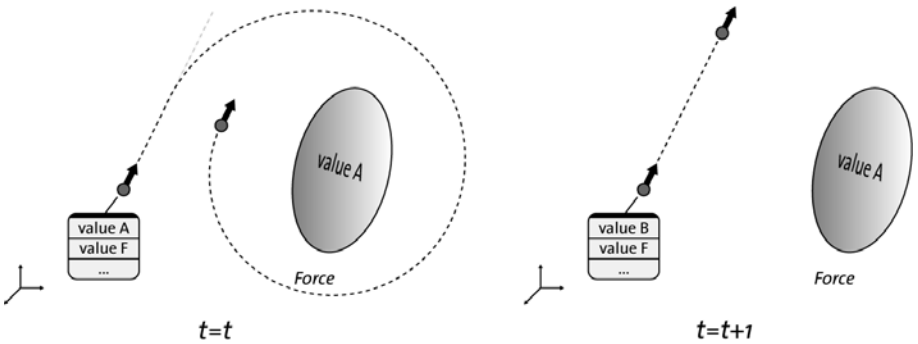


Fig. 13.4. A force attracts agents that contain data items with equal data values (left). Agents with dissimilar data items are not influenced by such a force (right).

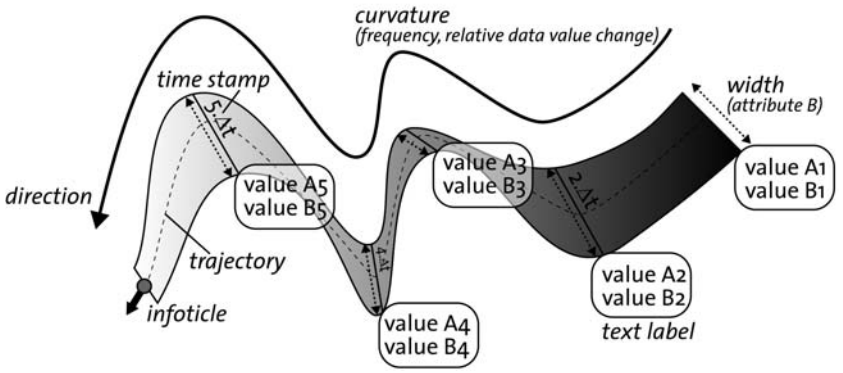


Fig. 13.5. Ribbon representation tracing the trajectory of the agent.

As shown in Fig. 13.5, flat ribbonlike visual elements graphically traced the spatial trajectories of the agents, so their dynamic behavior could be investigated in more detail, even in a static state. The width of the ribbon corresponds to a specific data attribute value, while time stamps clarify the historical trajectory of the agent.

Emergent Data Visualization Patterns

The infoticle method has been applied for a large dataset containing the Intranet file usage of a global, medium-sized company with about 7000 employees over a time frame of a year. Each agent represented a document that was stored on the Intranet file servers. The global offices were divided into seven different geographical categories (e.g., Asia, Europe, the United States, etc.), each of which was assigned a unique 'force.' The application timeline simulated the daily file usage according to the data stored inside the historical Intranet log files. During the visualization, Intranet files (or agents) traveled to those global offices in which they were downloaded for a specific time frame. As a result, the visualization conveyed the global knowledge distribution over time, so that particular data patterns could be detected and compared to the traditional means of Intranet log file reports and file usage rankings that the company was purchasing to assess the effectiveness of their Intranet. Several additional technical features dealt with the nature of continuous, widely ranging, unpredictable data streams of different time zones (Vande Moere et al. 2004).

Users could explore the data visualization at any moment by pausing, rewinding, and forwarding the application. Accordingly, agents moved smoothly back and forth along their historical trajectories. Because of the specifically designed interactions between speed and direction alterations, emergent visual effects appeared that conveyed meaningful data alteration trends over time. As agents could move freely in 3D space depending only on their attraction to a specific force, they were initially randomly distributed, so the initial point of origin of a data pattern plays no role. As illustrated in Fig. 13.6, the most meaningful emergent patterns included:

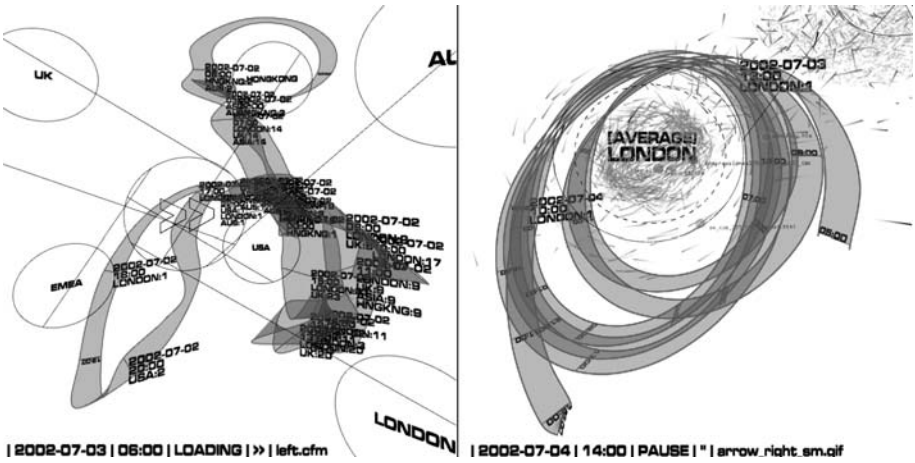


Fig. 13.6. Visually emergent infoticle patterns, resulting from data-driven self-organizing behavior. The circles represent the different geographical units; the ribbon and time stamps trace the three-dimensional trajectory of a particle. Visible patterns: erratic Quark pattern (left) vs. elliptical Comet patterns (right).

- *Star Pattern*. Documents that were only downloaded 'once' in a relatively long time frame 'circled' around their representative geographical forces. The smaller the agent-force distance, the longer ago they had been downloaded. These documents were old, were not effectively shared within the company, and possibly should be reclassified in the Intranet system.
- *Comet Pattern*. Documents that were downloaded often, but by the same geography, moved further away from that representative force in elliptical formed tracks. These were documents that were only shared within the same geographical unit, probably containing only locally valid knowledge.
- *Quark Pattern*. Documents that were downloaded (1) relatively often and (2) by several different regions were characterized by erratic movements in between the respective forces. Obviously, these particles represented the most shared documents, which contained information relevant to the whole company during that particular time frame.

Users could investigate the resulting visual patterns in a dynamic as well as static state. Dynamically, users specifically looked for erratic, circular, elliptical, or suddenly changing movements of agents, following the effect of specific document advertisements or the passing by of time-zone patterns. As illustrated in Fig. 13.7, users could search within spatial zones for regional trends or follow the performance of individual documents over time. In a static state, the formality of the ribbons tracing the agents' paths intuitively conveyed the dynamic characteristics of file usage. Globally, specific constellations of multiple agents appeared that conveyed so-called "axes of knowledge" between two or more dominant global offices.

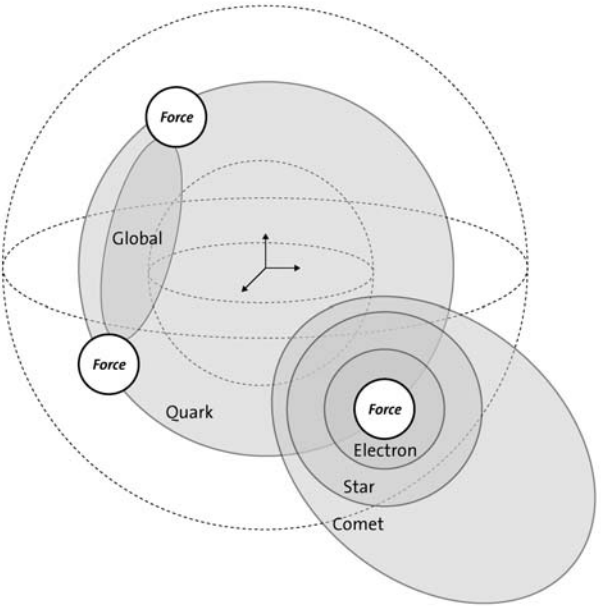


Fig. 13.7. Static zoning of agents according to the data pattern.

Several data patterns were discovered of which the company had not been aware. For instance, previously assumed highly 'popularity ranked' PDF documents tended to show up as comet patterns within very short time frames. Although those documents were originally reported to be "widely and popularly shared," their particular pattern proved that their relatively high download frequency was rather caused by a single user rapidly browsing through pages, causing the PDF software reader to cache separate pages and successively 'hit' the Intranet server.

13.4.2 Metaphor 2: Information Flocking Method

The information flocking method augments the previously described infoticle technique and its capability to simulate interpretable motion typologies, with the concept of *swarming*. Individual agents are enhanced with limited vision and communication capabilities, so they can identify other agents in their neighborhood, read their data items, and move toward or away from them without the need for external forces.

Self-Organization Method: Swarming

Swarming is based on the mathematical simulation of flocking birds or schooling fish. Models of flocking reveal that overall group structures in animals are directly affected by transformations at local levels (Couzin et al. 2002). More specifically, swarming is believed to reflect the underlying internal relationships and energy considerations

within the social hierarchy of a typical flock (Davies 2004). A possible functional explanation for emergent flocking behavior perceived in nature describes how animals at the edge of the herd are more likely to be selected by predators (Hamilton 1971). Accordingly, the flocking members would 'selfishly' attempt to move as close to the center of the herd as possible and, thus, in data visualization terminology, 'cluster' together. Reynolds (1987) successfully modeled the movements of so-called *boids* (or bird-objects) within a flock by designing a set of simple behavior rules that each member of the flock has to follow. This algorithm has been extensively used in the field of computer graphics, for instance, to simulate swarming in popular cartoon movies (Stanton and Unkrich 2003). The boid concept has also been used in a technical context, offering a mathematical model of dynamic formations with respect to parameters of wireless, ad hoc network communications systems (Kadrovach and Lamont 2002) or effective search strategies for performing exploratory geographical analyses (Macgill and Openshaw 1998). Swarming is an example of the so-called Particle-Swarm Optimization (PSO) research field which, like other evolutionary computation algorithms, can be applied to solve complex optimization problems from cross-disciplinary fields (Kennedy and Eberhart 2001).

Reynolds used observations of real flocks to derive the three primary behavior rules that each member of a flock needs to follow. Each of these rules is applied in parallel between all pairs of boids when the distance between them is smaller than predefined threshold values. Instead of simply averaging all behavior rules, a *prioritized acceleration allocation* strategy has to be used. This approach allocates priority according to the importance of the rule and normalizes the resulting values when a rule receives less than what was requested. This weighted average is used to compute the final velocity vector. The three behavior rules that govern each boid include (see Fig. 13.8):

- *Collision Avoidance.* Boids need to move away from other boids nearby, in order to avoid crashing into one another.
- *Velocity Matching.* Each boid should move with about the same speed as the agents in its neighborhood. As a result, multiple boids, as a group, seem to pursue a common goal.

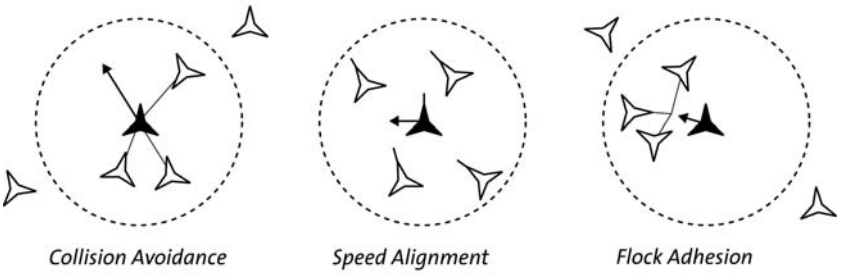


Fig. 13.8. Standard boid behavior rules (after Reynolds 1987).

- *Flock Centering.* Boids should attempt to stay close to other boids nearby. Boids in the center will feel no pull outward, whereas boids on the periphery will be deflected inward. As a result, each boid stays relatively near the center of the flock.

Swarming Simulation as Data Mapping

Flocking behavior has been explored for the purpose of information display by Proctor and Winter (1998). They showed how the clustering tendencies of schooling fish can be exploited to represent a pairwise similarity weight data matrix containing the relationships of interests to employees. We have extended this method with several features, such as the inclusion of time-varying data, a more stable flocking algorithm, and several technical features that are required to analyze the continuous stream of dynamic, unpredictable data in real time (Vande Moere 2004). Similar to the infoticle method, each agent is represented as a particle within 3D virtual space. However, the boid agents are able to sense the presence of other agents in their vicinity. Each agent is visually represented by a colored, curved line connecting the 3D points it has passed through and shows a gradually decreasing transparency to convey directionality. The agent color depicts positive (green) or negative (red) data value changes for a predefined data attribute. The agents' data items are sequentially updated according to the application timeline. In addition to the three traditional swarming rules mentioned earlier, each agent is governed by additional two behavior rules:

- *Data Similarity.* Each agent attempts to stay close to those agents with similar data values. This attraction rule results in the spatial clustering of agents that have similar data items. The similarity between agents is determined by testing whether the difference in data values of a single data attribute is lower than a predefined *data similarity threshold*.
- *Data Dissimilarity.* Each agent attempts to move away from those agents with dissimilar data values. This repulsion rule significantly accelerates the clustering effect of the first rule. Agents clustering too slowly would eventually cause the visualization to be confusing and nonrepresentative. Accordingly, the continuous progression of the timeline requires all boids to cluster, uncluster, and recluster efficiently.

It should be noted that too rapid data updates prevent a flock from evering attain a 'true equilibrium,' with boids finding their ideal position, and thus globally reaching a fully representative data representation. Therefore, it is necessary to stabilize the boid behavior as much as possible by fine-tuning the swarming weighting factor values and the pairwise boid threshold distances. The exact numerical values of these variables are determined through a trial-and-error process, as the application designer is unable to foresee the exact outcomes of the simultaneously applied local interactions.

Emergent Data Visualization Patterns

The information flocking method was tested with a dataset consisting of historical stock market quotes from one year totaling about 12,631 data entries ($\pm$500 companies

$\times \pm 200$ working days). The dataset was acquired from a public website (SWCP 2003) that accumulates historical stock market prices of the Standard & Poor's 500 Index, which represents a sample of 500 leading companies in the most important industries of the U.S. economy. Each data visualization agent represented a unique company. Agents calculated and compared the relative difference in price with their previous data update. As a result, the swarming behavior reflected the similarities in how their stock market quotes changed over time, rather than their exact stock market price. This method is capable of generating several interpretable emergent phenomena, as illustrated by Figs. 13.9 and 13.10.

- *Short-Term Clustering Patterns.* Boids that were near each other and moving in parallel directions represented similarities in stock quote change. Over time, several subclusters containing 'winning' and 'losing' companies formed within the main flock. Some individual boids and larger subflocks, consisting of 'outlying' agents that were affected by significantly different short-term stock market changes split away from the main flock.
- *Long-Term Zoning Patterns.* A visual distinction could be made between boids in the perceived flock center and those positioned in the flock periphery. Subsequent analysis of the corresponding stock market price evolutions showed that data items with long-term and volatile data value changes were loosely positioned at the outside of the flock, whereas relatively long-term stable conglomerate entities remained closely together in the flock center.
- *Collective Motion Behavior.* The main flock often suddenly changed its global direction or even seemed to 'implode' or 'explode' at specific moments in time. These erratic behaviors were traced back to significant global instabilities in the stock market. The emergent behavior can be explained by the 'guiding' influence

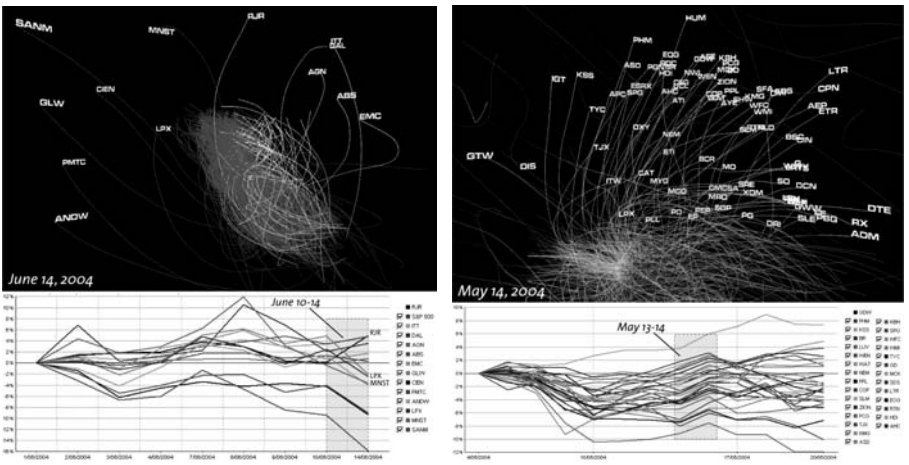


Fig. 13.9. Short-term zoning patterns: Individual boids being expulsed by the main flock (left), and subflocks with similar data value changes appearing from the main flock (right).

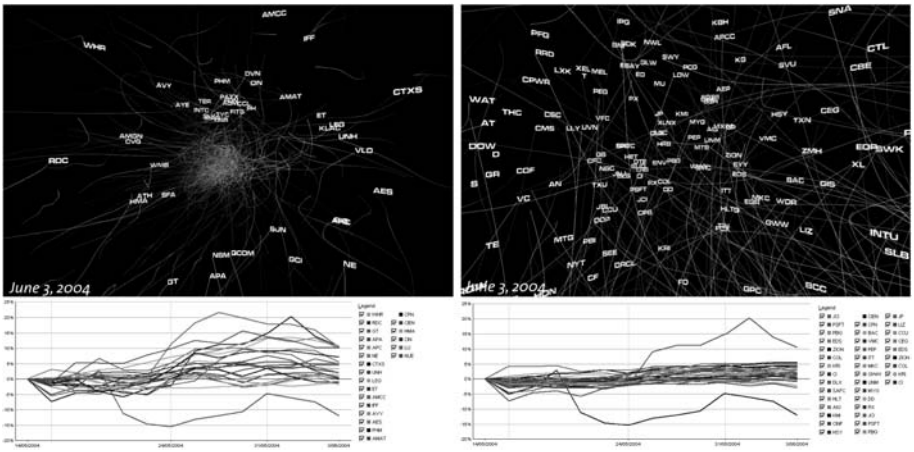


Fig. 13.10. Long-term zoning patterns: The flock core (left), vs. the flock periphery (right). The corresponding line graphs convey the relative volatility of the relevant stock quotes.

of the more volatile boids in the flock periphery, which join their position with boids in the flock center when their stock market quotes have stabilized, causing an apparent implosion or explosion.

- *Visual Swarming Formality.* It was expected that the formality of the boid swarm as a whole would correspond to specific data characteristics, so that, for instance, irregular formations denoted chaotic data alterations, while spherelike swarms formed out of stably altered stock market quote prices. However, it was concluded that such interpretations could only be made by a continuous observation of the changing shapes, and not from the perception of resulting static formations.

The information flocking method is able to represent short-term (e.g., clustering) as well as long-term (e.g., zoning) data tendencies in datasets with complex and potentially randomly changing data values. It is capable of displaying hundreds of time-varying data items simultaneously, even when the data changes in real time, as it performs the data value comparisons within the visualization canvas itself. Displaying or even visually analyzing such a number of stock market companies simultaneously via line graphs would be impracticable. Whereas this approach is certainly not ideal for real-world stock exchange applications, it is proposed as a prototype toward more usable visualizations that are able to perform data analysis as well as data representation simultaneously and in real time. It also illustrates how motion typology and spatial clustering can be used to convey complex data tendencies.

13.4.3 Metaphor 3: Cellular Ant Method

The *cellular ant* method augments the previously applied principles of self-organization. Agents will also be able to determine their visual attributes, such as its position, color, and shape size. The resulting visual diagrams appear similar to those generated by the *multidimensional scaling* (MDS) approach, which displays the structure of

distancelike datasets as simple geometrical pictures (Torgerson 1952) arranged in 2D space. In MDS diagrams, the distance between pairs of data items denotes the degree of data similarity. Several MDS-like data visualization techniques exist in combination for instance, with animation (Bentley and Ward, 1996) or recursive pattern arrangements (Ankerst et al. 1998). One should note that multidimensional scaling differs from clustering in that the latter partitions data into classes, whereas MDS computes positions without providing an explicit decomposition into groups.

Self-Organization Method: Cellular Automata and Ant Foraging

The cellular ant method merges two established approaches: *ant-based foraging* from applied artificial intelligence and *cellular automata* from artificial life. More particularly, agent-based foraging is derived from the nest-cleaning characteristics of ant colonies in nature and has been successfully applied to data clustering in the field of data mining.

Cellular automata (CA) constitute a well-known computational method originally proposed by Von Neumann (1966). Which consists of a number of cells in a grid each representing a discrete state (i.e., 'alive' or 'dead'). All cells are governed by behavior rules that are iteratively applied and generally only consider the states of the neighboring cells. When a specifically designed rule set is applied on well-chosen initialization constellations, visually intriguing cell patterns can occur emergently, such as demonstrated by the Game of Life application invented by Conway (see Gardner 1970). Cellular automata can also be used for a case study to investigate the controlling mechanisms of large multiagent systems (Zambonelli et al. 2002).

Ant foraging is an example of *ant-based data mining*, which in turn combines the nest-cleaning characteristics of ant colonies with the task of data clustering. The typical ant-clustering algorithm starts from a toroidal grid on which data objects are randomly scattered, and are thus considered 'lifeless' entities that are moved around by foraging ants (Deneubourg et al. 1990; Lumer and Faieta 1994; Kuntz et al. 1997). Ants pick up data items and move around in randomly chosen directions. They then probabilistically decide whether to drop the data item, preferably in the vicinity of similar data items. A specific *object distance measure* variable α determines the degree of similarity between pairs of data objects, so that dissimilar items will not be placed together and similar items will be clustered. The optimal value for α cannot be determined without prior knowledge of the dataset, unless its value is adaptable (Handl and Meyer 2002). There are different ant-based clustering approaches incorporating fuzzy-set theory (Pham and Brown 2004), topographic maps (Handl et al. 2005), or even biologically inspired genetic algorithms (Ramos and Abraham 2004). These ant-based data mining techniques typically result in graphical diagrams with spatially separate clusters that, internally as well as relatively to each other, are unordered in terms of data value. In effect, the usefulness and effectiveness of these methods for visual information display is low.

While CA simulations are typically determined by environmental states, ants can 'act' upon the environment and even change it to some degree, for instance, by removing and dropping objects. The cellular ant approach applies typical CA rules,

traditionally used for grid cell states, to the perception and reasoning of ants: cellular ant agents decide their actions depending on the discrete number of 'similar' agents in their neighborhood, rather than using probabilistic mathematical functions. The essence of the cellular ant methodology is thus that agents will roam around, a technique similar to the ant-based foraging technique, and take actions depending on neighborhood densities, similar to the cellular automata technique. In addition, instead of simply picking up data objects, the ants move, as they 'are' the data items themselves. The idea of mapping data objects directly onto ants is relatively new. Labroche et al. (2002) associates data objects with ant genomes and simulates meetings between them to dynamically build partitions: ants detect the label that best fits their genome, which corresponds to the best cluster. A recent data mining clustering method has shown that data objects can be mapped onto ants, of which the apparent behaviors resemble that of cellular automata (Chen et al. 2004). It differs from the proposed agent data visualization methodology as it does not spatially order clusters and is based on probability functions and predefined internal ant states.

Cellular Automata Ants as Data Mapping

Each data visualization agent is represented as a unique colored square cell, positioned within a toroidal rectangular grid. Each agent has a limited perception of its surrounding neighborhood, which consists of eight neighboring cells (see Fig. 13.11). Similarly to the swarming approach, an agent's behavior is based on a set of behavior rules that takes into account the presence of all other agents in its local neighborhood and their corresponding data values. However, agents are in control of their dynamic behavior as well as their visual properties; each agent can move around or stay put, swap its position with a neighbor, and adapt its own color or negotiate its shape size.

At initialization, all agents are randomly positioned within a toroidal grid. Similar to the classical MDS (or CMDS) method, each agent calculates the Euclidian distance between its own normalized data item and that of each of its eight neighbors. This data distance measure represents an approximation of the similarity between pairs of data items, even when they contain multidimensional data values. Next, an agent will only consider and summate those agents for which the pairwise similarity distance is below a specific, predefined *data similarity tolerance threshold value t*. Value t is conceptually similar to the *object distance measure* α in common ant-based clustering

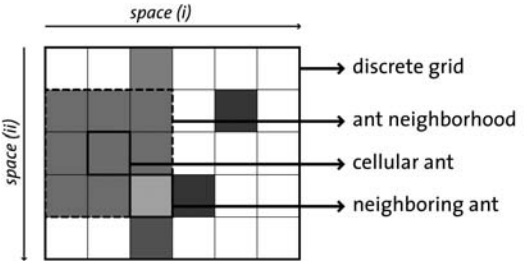


Fig. 13.11. Data visualization agent (or cellular ant) within its grid cell neighborhood.

approaches. However, t originates from a cellular automata approach in that it is a fixed and discrete value, which generates a Boolean result (i.e., a pair of data objects is either 'similar' or not) instead of a continuous similarity value (e.g., representing a numerical degree of similarity between pairs of data objects). Depending on the number of agents in its neighborhood a particular agent considers as 'similar,' an agent will then decide either to stay put or to move. For each iteration, each agent is governed by five different behavior rules: surface tension and edge repulsion (together causing clustering), positional swapping, color determination, and shape size adaptation (Vande Moere and Clayden 2005; Vande Moere et al. 2006):

- *Surface Tension.* Each agent has the freedom to roam around and meet other agents. If an agent has less than four similar neighbors, it will attempt to move to an empty cell next to the most similar ant in its neighborhood. Once an agent has four or more similar neighbors, it will stay put. As a result, agents that represent similar data items will stay close to each other, and small clusters form that act as seeds for other agents to connect to. Over time, these initial clusters can grow or merge with each other.
- *Edge Repulsion.* An agent that has one or more dissimilar agents in its neighborhood moves away from its current position, toward an empty cell closest to its most similar neighbor. This part of the algorithm is meant to create series of 'empty cells' as 'borders' around the separate clusters, so geometric clusters become better visually distinguishable. This means that two or more individual clusters that eventually 'share' a border with each other contain similar data elements, thus creating a more meaningful data representation.
- *Positional Swapping.* At each iteration, each agent picks a random direction (i.e., horizontal, vertical, or one of both diagonals) in its neighborhood, with itself as the middle agent. It then reads the normalized data values of the corresponding neighbors, and calculates the (one-dimensional) distances between all these agents in the multidimensional parameter space. Based on these three numerical pairwise values, an agent is able to determine if it needs to 'swap' its position with one or both of its outer neighbors, or whether the current constellation is ideal. In practice, this means that if the Euclidian distance in parameter space between the middle agent and one of the outer agents is greater than the distance between the outer agents, the middle agent has to swap with the most similar outer agent (see Fig. 13.12). When agents swap, the data value gradient between the three agents becomes continuous and monotonic. Subsequently, the swapping rule linearly orders agents in any chosen grid direction by multidimensional data similarity, so that 'more similar' agents are positioned closer to each other and dissimilar ones are placed further apart in the grid. Although this rule is applied in randomly chosen directions, a globally ordered structure emerges due to the multitude of iterations.
- *Color Determination.* At initialization, all agents are assigned a neutral color (i.e., white). Each ant that has not been swapped (and thus probably is well placed within its neighborhood) and is fully surrounded by eight similar neighbors considers the degree of data similarity with all of its neighbors. If this degree is below a predefined, discrete *color seed similarity threshold c*, it will request a unique color from the application system. An agent with a neighborhood containing four or more data

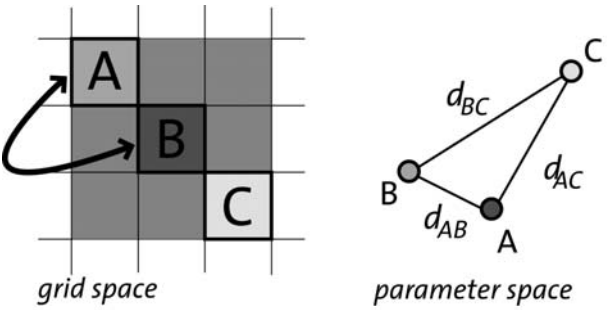


Fig. 13.12. Agents swapping grid position depending on data value. The largest data value distance in parameter space is d_{BC}, so that agent A should be positioned between B and C.

items of which the Euclidian distance is smaller than t but greater than c is relatively 'satisfied' with its current position and adopts the color of the most similar agent in its neighborhood. As a result, once the collection of ants is sufficiently ordered, colors are introduced where there are stable clusters of very similar ants. The individual agents that have received a unique color act as initial 'color seeds,' which spread throughout the whole agent population. Because of the multitude of pairwise interactions, any surplus of colors (in respect to data clusters) will disappear, while any shortfall of colors will reemerge once another potential ant is surrounded by eight very similar neighbors. One should note that the color of an agent is thus identical to its assumed *data class* or *data label.* Consequently, the resulting diagrams resemble that of (ant-based) data clustering in the domain of unsupervised data mining.

- *Shape Size Determination.* For each iteration step, the visual shape size of an ant is determined by the following inducements. First, an agent chooses a random neighboring agent and reads the numerical value of a predefined data attribute and its circular radius size, measured in screen pixels. It then considers whether its own radius vs. data value ratio is similar to that of the neighboring agent and adapts its own as well as its neighbor's shape size accordingly. For instance, if its size is too large in relation to its data value, it 'shrinks' by decreasing its amount of available pixels by a specific number of pixels and then offers these pixels to the other agent. When an agent becomes too large, it will 'punish' and decrease the size of its neighbor, so that this step will not be required in the next iteration. These constraints will emergently 'detect' the upper and lower shape size boundaries according to the data value scale, which emergently spread throughout all ants. Accordingly, instead of mapping data values directly onto specific shape sizes, each agent is able to map one of its data attributes onto its size by negotiating with its local neighbors. However, the size of an agent does not necessarily correspond to the 'exact' value of that data attribute, but rather to how a data value locally relates to its neighborhood and, therefore, whether clusters are locally homogeneous in respect to a specific data attribute. The shape size scale is able to autonomously adapt to the data scale, without the need for a separate data analysis or a predefined mapping rule between assumed data values and the visual transformation (see also Sec. 13.3.1).

Emergent Data Visualization Patterns

The cellular ant method was applied with various artificially created and standard machine learning benchmarking datasets, ranging from two to up to nine data dimensions, and from 150 to 768 data items. Several emergent effects can be perceived:

- *Clustering.* As shown in Fig. 13.13, the initially randomly scattered data items self-organize in clearly perceivable clusters of similar data type over a process of several hundred iterations. The number of required iterations varies with the random initialization distribution and with the grid size for an equal dataset. On an emergent level, a relatively long period of iterations in apparent chaos suddenly transforms into an orderly and stable constellation.

- *Internal Cluster Order.* As shown in Fig. 13.14, the resulting clusters themselves are ordered internally: data items that are similar in parameter space are also positioned near each other in visualization space. For instance, the highlighted ants share a border between different adjoining clusters, but also share this border in the 2D parameter space. This particular emergent phenomenon is ideal for the purpose of information display, as the resulting clusters have meaning and can be easily interpreted by users.

- *Relative Cluster Order.* The clusters themselves, seen as global phenomena, are positioned in an ordered way within the visualization grid. Cluster that are dissimilar in parameter space have no 'common' borders in visualization space, and vice versa. For instance, the diagonal clusters in parameter space do not share any border in visualization space, as they are more dissimilar than the horizontally and vertically adjoining clusters. Accordingly, the information display is more accurately representative and more easily comprehensible in comparison with other unsupervised clustering methods.
- *Data Class Determination.* The color determination rule results in a few different colors emergently appearing. Each color corresponds to a separate data class or data category consisting of multiple data items that are significantly similar. At the start, many colors appear, of which most disappear by merging with on another. Figure 13.15 shows how a car specification dataset [containing 38 items and 7 data dimensions, as taken from Wojciech (2001)] results in a single spatial cluster, with three different colors. Whereas the spatial clustering was not able to separate the

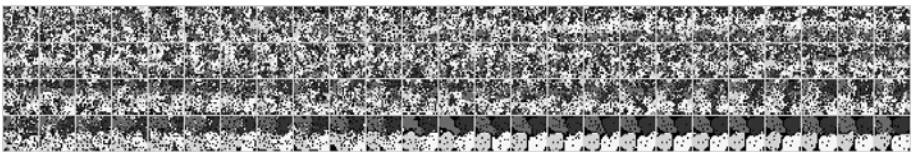


Fig. 13.13. Timeline snapshots of a random constellation of agents clustering, one snapshot per 100 iterations.

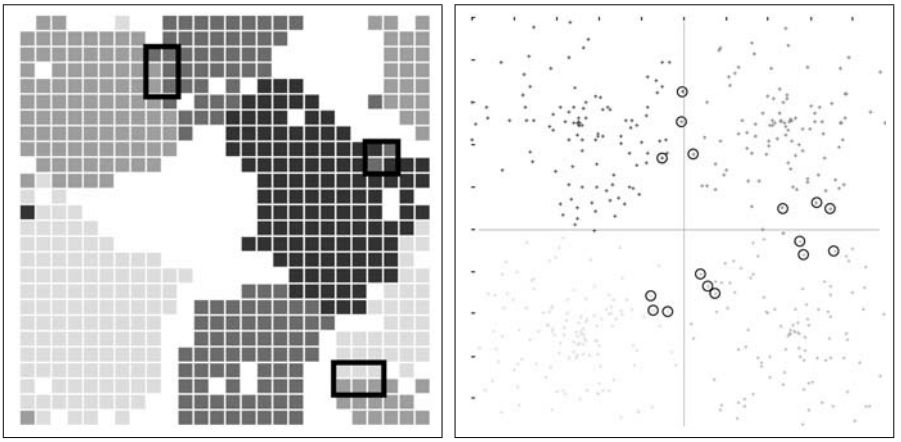


Fig. 13.14. Internal cluster order according to data similarity (500 ants, toroidal 26×26 grid, 2D synthetic dataset, four classes, $t = 0.27$). Agents positioned in grid space (left) vs. data items in 2D parameter space (right). The highlighted grid cells and parameter points demonstrate that agents on 'shared' borders of two clusters are also shared in parameter space. Respective diagonal clusters in parameter space do not share any borders in the visualization.

different types of cars because of similarity links between them, the color negotiation recognized three different types, roughly corresponding to the number of car cylinders. The color negotiation is another feature that increases the legibility of the resulting information display.

- *Relative Data Attribute Distribution.* Figure 13.16 illustrates how shape size negotiation can be used to clarify high-dimensional data dependencies, without prior knowledge of the data scale and without using any predefined data mapping rules. Figure 13.16 uses the same dataset as Fig. 13.15, but maps a single data attribute to circle size, showing the relative dominance of the cylinder count and miles per gallon within clearly distinguishable clusters. The shape negotiation allows users to investigate how different singular data attributes are relatively distributed within multidimensional clusters.

One should note that the effectiveness of the cellular ant method is influenced by several variables that need to be configured beforehand and depends primarily on the dataset characteristics. The influential parameters include specific threshold values and overall grid size. Their specification requires a process of trial-by-error iterations to ensure that all the variables are optimally chosen. Also, it cannot be guaranteed that this method will always finish with a stable emergent result: a small number of ants might be swapping indefinitely, or individual ants (e.g., data outliers) might continuously roam around. Therefore, a simple performance graph was implemented that summated the number of similar ants in each ant's neighborhood. The visualization's efficiency is indicated by the slope of the corresponding graph. Once a plateau value

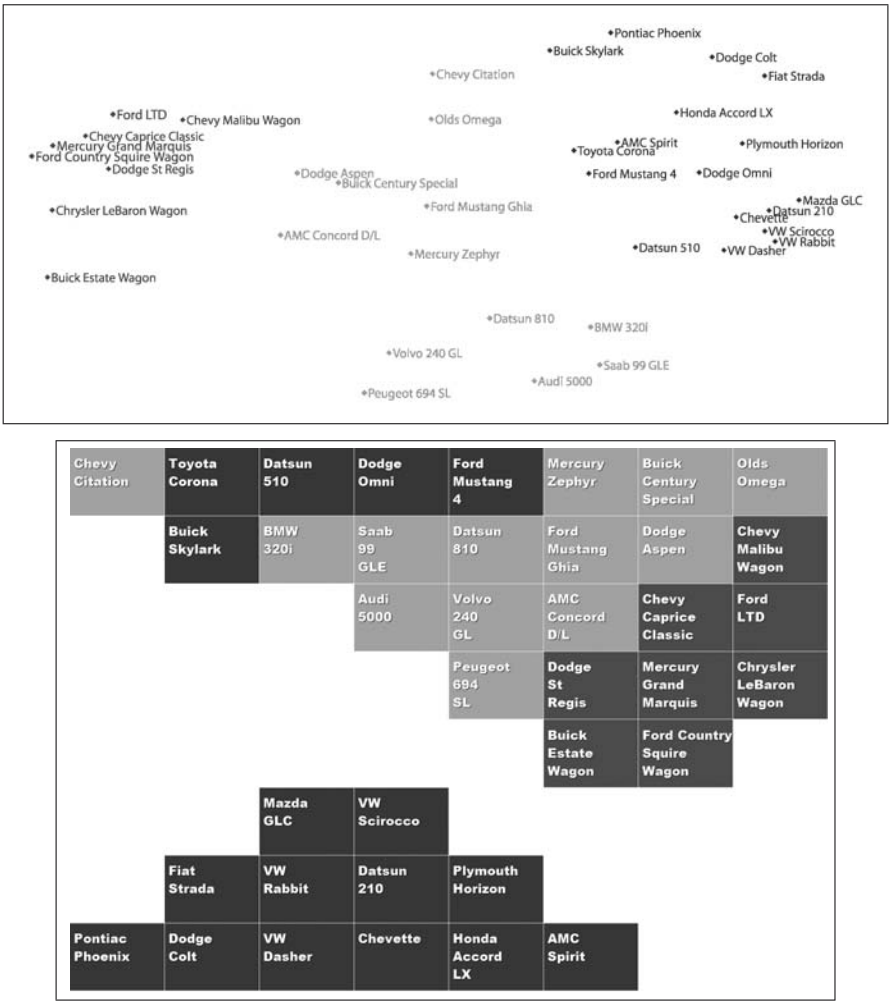


Fig. 13.15. Two different data visualizations of a car dataset. Top: represented by traditional MDS [diagram based on Wojciech (2001), any grayscaling added by hand]; Bottom: represented by the cellular ants representation with unsupervised color negotiation, shown here in grayscale.

has been reached over a number of iterations, the visualization has reached a stable state and can be halted. Quantitative benchmarking has shown similar performance with comparable data mining techniques (Vande Moere and Clayden 2005; Vande Moere et al. 2006). This simple prototype was developed to demonstrate that even simple self-organizing principles can lead to useful and meaningful results. Future work should focus on conceptual optimizations that should allow the method to be less dependent on external variables and increase the overall performance in terms of required iterations, calculation performance, and configurability.

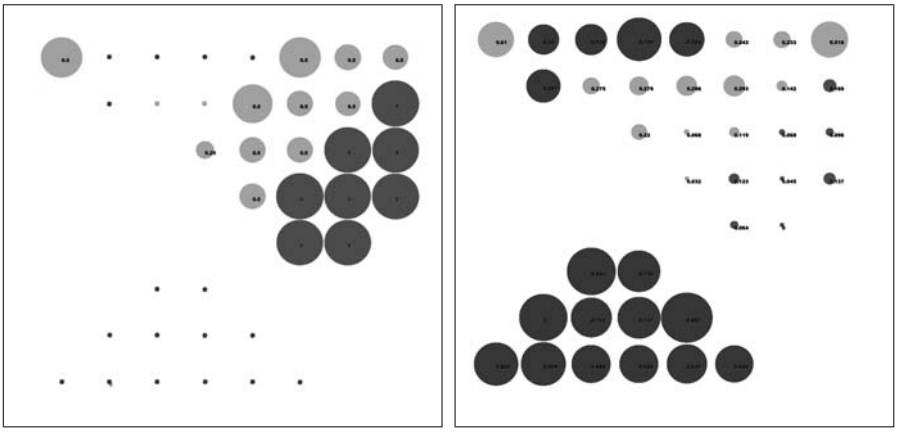


Fig. 13.16. A cellular ant representation in a toroidal grid with color and shape negotiation. Singular data attribute represented by the shape size: cylinder count (left) and miles per gallon (right).

13.5 Discussion

Data visualization supports users in creating a mental model by artificially constructing a visual representation from otherwise invisible data characteristics. An effective visualization facilitates rapid understanding of this model, generating intuitive connotations between visually emergent patterns and meaningful data patterns. This section discusses some of the most serious shortcomings of the self-organizing method and some potential future solutions.

13.5.1 Analysis

The unique characteristics of the self-organizing data visualization model is related to a set of specific issues, including:

- *Real-Time Data Analysis.* The self-organizing approach calculates data similarities in real time. This means that a user does not necessarily need to conduct a pre-analysis of the datasets to determine the upper and lower bounds of the data mapping and that datasets even can be updated by external sources in real time. However, the continuous streaming of time-varying data poses specific issues as data items tend to be updated in irregular sequences so that some degree of data time averaging is required. In addition, the continuous dataset querying, networking, and analysis computations can negatively influence the calculation performance.
- *Calculation Performance.* As with any *parallel algorithm*, the simultaneous execution of behavior rules for each single agent is relatively calculation intensive. This effort increases exponentially with the number of agents and thus with the dataset size. Especially for the information for the flocking method, which requires one-to-one comparisons, and the cellular ant method, which requires one-to-eight comparisons, the dataset size is an important delimiting factor. Possible solutions for

optimizing this issue include: (1) predefined data similarity lookup tables, so that data item comparisons do not need to be calculated on the fly; (2) skipping behavior rules for specific iteration cycles, so that agents are calculated progressively; or (3) giving priority to specific agents, for instance, those that are recognized to be guiding the emergent process. More technical solutions for performance optimization consist of (4) real-time data optimization, including data approximation and gradual data streaming (Hellerstein et al. 1999), as well as agent adaptation; (5) the distribution or balancing of loads between agents (Decker and Sycara 1997); and (6) agent cooperation, by creating coalition formations of 'superagents' that adapt together as they have similar objectives, experience, or goals (Ogston et al. 2003).

- *Nondeterministic Results.* Using emergence as the core organizing principle inevitably ports aspects of uncertainty in the final results. This means that most generated representations are *nondeterministic*: different graphical diagrams can result from equal datasets owing to different initialization conditions. However, recurring data trends are consistently represented. In the case of the cellular ants, different initialization constellations inevitably result in different amounts of clusters, colors, and sizes, so that multiple program executions and averaging are required to ensure results that can be confidently generalized.
- *Equilibrium.* Because of the continuous nature of the iterative cycles, it is often not intuitively determinable when the representation is 'finished,' i.e., when the agents have found a suitable equilibrium for the current dataset situation. Therefore, self-organizing methods require some externally measurable degree of agent 'satisfaction,' determined holistically over the whole agent population. Once this measurement reaches a 'plateau' over time, one can assume that the optimal solution for a specific configuration has been reached.
- *Comprehensibility*. The three different case studies have demonstrated how motion typology or visual properties of particles and colored grid cells are able to convey useful data patterns. However, users are still required to correlate and interpret these behavioral effects to particular meanings in relation to the dataset. Users might initially be confused because of the reliance on fully dynamic features, which are typically underused in most software applications. There is also a latent danger of allowing users to interpret some emergent effects as meaningful, whereas no cause and effect in the context of the dataset could be detected.
- *Parameter Influence.* As with any algorithm that applies principles of self-organization, the occurrence of emergence is highly reliant on fine-tuning the variables involved. Altering these variables tends to result in different emergent visual effects, thereby influencing the effectiveness of depicting specific data patterns. Some variables control the visual effect of the emergent phenomena, so that they become easier to perceive and understand. As shown in Table 13.1, in the case of information flocking, at least 10 different variables are required to drive the traditional flocking behavior alone: three traditional plus two data similarity variables determine the Euclidian distances at which the behavior rules need to be invoked, and another three traditional plus two data similarity variables specify the relative importance of each behavior rule to one another (Reynolds 1987). The exact, ideal specification of such emergent-behavior-influencing variables gener-

ally requires a relatively large number of *trial-and-error* iterations, as one is unable to predict the emergent behavior of hundreds or thousands of internally interacting elements. Additional variables fine-tune the data analysis process: for instance, for each dataset separately, the exact threshold value that determines whether two data items are 'similar' or not needs to be defined. This value can be found by measuring the performance of agents toward a global equilibrium over several iterations. Several solutions for this problem exist, such as the implementation of an easily configurable software framework that allows the developer to fine-tune the configurable variables on-the-fly, without the need to change any programming code. Alternatively, the software itself could determine and then 'optimize' the possible solution space by brute force experimentation and present the results to the user. Lastly, most of the variables could gradually change over time, toward their assumed optimal value, allowing the emergent processes to be influenced by alternative values in a dynamic way.

- *Dataset Characteristics.* The different case studies have proven how self-organization for data visualization purposes can be accomplished with relatively small and low-dimensional datasets, as the research focused on determining the feasibility of the proposed methodology. In particular, this technique seems to be ideally suited for relatively small, middimensional, and time-varying datasets. Further research is needed to determine whether this approach maintains its usefulness for more complex datasets, in terms of size, dimensionality, or time variance.

Table 13.1 summarizes the qualitative differences among the three case studies. As these different applications targeted fundamentally different datasets, purposes, and usages, no benchmarking comparisons have been made.

13.5.2 Future

Traditional data visualization applications rely on the discretion and expertise of users in detecting data patterns. Pattern detection is complex, and therefore a future potential exists for using the results of data analysis to 'inform' the data mapping process. It is in this aspect that self-organizing data visualization might have the most useful potential: to construct a more effective, and more informed, visual representation based on the unsupervised analysis of dataset properties, visualization guidelines, and visual cognitive knowledge. Such an approach would be the first step toward more intelligent data visualization applications that become 'aware' of the data they represent, the useful patterns they contain, and the most effective techniques to translate these into visual form. By using principles of self-organization, the intelligence required to detect data patterns is distributed by spreading the required computations over multiple equal elements. As a result, the application can analyze the data in real time, becomes tolerant of local failures, allows for unexpected data alterations, and potentially creates emergent solutions that are optimized to the specific patterns in and characteristics of the dataset. A possible future for self-organizing data visualization could lead to data mappings that are themselves emergent, creating unique, unforeseen, but ordered and interpretable representations that inherently reflect the relationships and complexities hidden within the dataset.

Table 13.1. Qualitative differences between the 3 case studies demonstrating the self-organizing data visualization model.

	Infoticle	Information flocking	Cellular ants
Agent concept	Particle	Boid	Antlike grid cell state
Agent description	A point in 3D space that is attracted by a set of point forces following the rules of Newtonian mechanics	A point in 3D space that swarms or flocks with similar flock members	A grid cell that can move to neighboring grid cells, similar to the movement of a ant
Dataset features	Time-varying Two dimensional	Time-varying Two dimensional	Static Multidimensional
Behavior algorithm	Rule based Cause and effect	Rule based Decentralized interagent communication	Rule based Decentralized interagent communication
Behavior influence	Data alterations over time Position of force in space	Data alterations over time Pairwise data item comparisons	Data item comparisons among up to eight neighbors
Visual data mapping cues	3D dynamic motion, determined by speed and direction	3D dynamic motion, determined by speed and direction Motion typology	2D position by one-cell movement and pairwise swapping Color Size
Emergent phenomena	Individual motion typology Grouping by collective motion typology Spatial grouping Motion path formality	Spatial clustering (position, speed, and direction) Spatial zoning Individual and collective motion typology	Clustering Size determination Color determination
Emergence variables	Speed alteration (2) Force data attributes (1)	Flocking rules neighborhood distances (5) Prioritized acceleration allocation parameters (5) Data similarity threshold t	Data similarity threshold t Color seed threshold c Square grid size (1)

Self-organizing data visualization evokes visual emergence by behavioral emergence. It is able to determine the position, color, shape, and dynamic behavior of a data item according to its inherent relationships with other data items in the dataset. Each data item itself is a member of a decentralized multiagent system, based on existing emergent simulations such as particle animation, flocking and swarming, ant-foraging, or cellular automata. Self-organizing data visualization exploits already known emer-

gent simulation algorithms to generate data-driven visual patterns. It is based on the assumption that a dataset is a collaborative agent system, which inherently contains all the knowledge required to derive a useful visual representation. Such a dataset is capable of determining its own visual presence and conveying the most meaningful data trends that it contains. Therefore, the self-organizing data visualization model points to a future in which data visualization becomes a data analysis tool that is aware of the data it represents and how it should adapt its representation accordingly. As discussed earlier, several technical and conceptual hurdles still need to be overcome before principles of self-organizing data visualization can be integrated into useful applications. However, such applications will be able to proactively support users in their continuous quest to understand the ever-more complex data collections that are continuously generated and accumulated in our society.

References

Ankerst, M., Berchtold, S., and Keim, D. A. (1998). Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proceedings of Symposium on Information Visualization (Infovis'98)*, pages 52–60, IEEE Computer Society, Washington DC, USA.

Bartram, L., and Ware, C. (2002). Filtering and brushing with motion. *Journal of Information Visualization*, 1(1): 66–79.

Bentley, C. L., and Ward, M. O. (1996). Animating multidimensional scaling to visualize N-dimensional data sets. In *Proceedings of Symposium on Information Visualization (Infovis'96)*, pages 72–73, IEEE Computer Society, Washington DC, USA.

Brodlie, K. W., Brooke, J., Chen, M., Chisnall, D., Hughes, C. J., John, N. W. (2006). A framework for adaptive visualization. *IEEE Visualization 2006*, Baltimore, MD, IEEE Computer Society, Washington DC, USA.

Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, San Francisco.

Chen, L., Xu, X., Chen, Y., and He, P. (2004). A novel ant clustering algorithm based on cellular automata. In *Proceedings of International Conference of the Intelligent Agent Technology (IAT'04)*, pages 148–154, IEEE Computer Society, Washington DC, USA.

Chi, E. H. (2000). A taxonomy of visualization techniques using the data state reference model. In *Proceedings of IEEE Symposium on Information Visualization (Infovis)*, pages 69–75.

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., and Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218:1–11.

Davies, J. (2004). Why birds fly in formation: A new interpretation. *Interpretive Birding*, 5(2).

Decker, K. S., and Sycara, K. (1997). Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9(3):239–260.

Deneubourg, J., Goss, S., Franks, N., A., S. F., Detrain, C., and Chretian, L. (1990). The dynamics of collective sorting: Robot-like ants and ant-like robots. *From Animals to Animats: 1st International Conference on Simulation of Adaptative Behaviour*, pages 356–363.

Ebert, A., Bender, M., Barthel, H., and Divivier, A. (2001). Tuning a component-based visualization system architecture by agents. In *Proceedings of International Symposium on Smart Graphics*, IBM T.J. Watson Research Center.

Eick, S. G. (2001). Visualizing online activity. *Communications of the ACM*, 44(8):45–50.

Franklin, S., and Graesser, A. (1996). Is it an agent, or just a program? A taxonomy for autonomous agents. In *Proceedings of Third International Workshop on Agent Theories, Architectures, and Languages (ATAL'96)*, pages 21–35, Springer, Heidelberg.

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life.' *Scientific American* 223:120–123.

Hamilton, W. D. (1971). Geometry for the selfish herd. *Journal for Theoretical Biology*, 31: 295–311.

Handl, J., and Meyer, B. (2002). Improved ant-based clustering and sorting in a document retrieval interface. In *Proceedings of International Conference on Parallel Problem Solving from Nature (PPSN VII)*, volume 2439 of *Lecture Notes of Computer Science*, pages 913–923, Springer, Heidelberg.

Handl, J., Knowles, J., and Dorigo, M. (2005). Ant-based clustering and topographic mapping. *Artificial Life*, 12(1):35–61.

Healey, C. G., Amant, R. S., and Chang, J. (2001). Assisted visualization of E-commerce auction agents. In *Proceedings of Graphics Interface 2001*, Ottawa, Canada, pages 201–208, Lawrence, Erlbaum Associate, New Jersey.

Hellerstein, J. M., Chou, A., Hidber, C., Olston, C., Raman, V., Roth, T., et al. (1999). Interactive data analysis: The control project. *IEEE Computer*, 32(8):51.

Hiraishi, H., Sawai, H., and Mizoguchi, F. (2001). Design of a visualization agent for WWW information. Volume 2112 of *Lecture Notes in Computer Science*, pages 249–258, Springer, Heidelberg.

Ishizaki, S. (1996). Multiagent model of dynamic design: Visualization as an emergent behavior of active design agents. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems (CHI'96)*, Vancouver, British Columbia, Canada, pages 347–354.

Jankun-Kelly, T. J., Ma, K.-L., and Gertz, M. (2002). A model for the visualization exploration process. In *Proceedings of IEEE Visualization*, Boston, IEEE Computer Society, Washington, DC, USA.

Kadrovach, B. A., and Lamont, G. B. (2002). A particle swarm model for swarm-based networked sensor systems. In *Proceedings of ACM Symposium on Applied Computing, Madrid, Spain*, pages 918–924, ACM, New York.

Kennedy, J., and Eberhart, R. C. (2001). *Swarm Intelligence*. Morgan Kaufmann, San Francisco.

Kramer, P., and Yantis, S. (1997). Perceptual grouping in space and time: Evidence from the Ternus display. *Perception and Psychophysics*, 59(1):87–99.

Kuntz, A., Layzell, P., and Snyers, D. (1997). A colony of ant-like agents for partitioning in VLSI technology. In *Proceedings of European Conference on Articial Life*, pages 417–424. The MIT Prss, Cambridge.

Labroche, N., Monmarché, N., and Venturini, G. (2002). A new clustering algorithm based on the chemical recognition system of ants. In *Proceedings of European Conference on Artificial Intelligence, Lyon, France*, pages 345–349, IOS Press, Amsterdam.

Lander, J. (1998). Ocean spray in your face. *Game Developer*, 9–13.

Lethbridge, T. C., and Ware, C. (1990). Animation using behavior functions. In T. Ichikawa, E. Jungert and R. R. Korfhage, editors, *Visual Languages and Applications*, pages 237-252. Plenum Press, New York.

Lumer, E. D., and Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. *From Animals to Animats: Conference on Simulation of Adaptative Behaviour*, pages 501–508, The MIT Press, Cambridge.

Macgill, J., and Openshaw, S. (1998,). The use of flocks to drive a geographic analysis machine. In *Proceedings of International Conference on GeoComputation, Bristol, United Kingdom*. GeoComputation CD-ROM, Manchester.

Mackinlay, J. D. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2): 110–141.

Marefat, M. M., Varecka, A. F., and Yost, J. (1997). An intelligent visualization agent for simulation-based decision support. *Computing in Science and Engineering*, 4(3), 72–82.

Martin, A. (1999). Particle systems. retrieved August, 2006, from http://www.cs.wpi.edu/∼matt/courses/cs563/talks/psys.html

Mason, K., Denzinger, J., and Carpendale, S. (2004). Negotiating Gestalt: Artistic expression and coalition formation in multiagent systems. In *Proceedings of International Symposium on Smart Graphics*, pages 103–115.

Ogston, E., Overeinder, B., van Steen, M., and Brazier, F. (2003). A method for decentralized clustering in large multi-agent systems. In *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia*, pages 789–796, ACM, New York.

Pham, B., and Brown, R. (2003). Multi-agent approach for visualisation of fuzzy systems. Volume 2659 of *Lecture Notes in Computer Science*, pages 995–1004, Springer, Berlin.

Proctor, G., and Winter, C. (1998). Information flocking: Data visualisation in virtual worlds using emergent behaviours. In *Proceedings of Virtual Worlds 98, Paris, France*, pages 168–176, Springer, Berlin.

Ramos, V., and Abraham, A. (2004). Evolving a stigmergic self-organized data-mining. In *Proceedings of International Conference on Intelligent Systems, Design and Applications (ISDA-04), Budapest, Hungary*, pages 725-730, IEEE Computer Society, Washington, DC, USA.

Reeves, W. T. (1983). Particle systems: A technique for modeling a class of fuzzy objects. *Computer-Graphics*, 17(3):359–376.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Roard, N., and Jones, M. W. (2006). Agent based visualization and strategies. In *Proceedings of Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Pilzen, Czech Republic*. University of West Bohemia, Plzen.

Robinson, N., and Shapcott, M. (2002). Data mining information visualisation: Beyond charts and graphs. In *Proceedings of International Conference on Information Visualisation, London*, pages 577–583.

Sadok, B. Y., and Engelbert, M. N. (2004). Emulating a cooperative behavior in a generic association rule visualization tool. In *Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04), Washington, DC, USA*, pages 148–155, IEEE Computer Society, Washington, DC, USA.

Schroeder, M., and Noy, P. (2001). Multi-agent visualisation based on multivariate data. In *Proceedings of International Conference on Autonomous Agents, Montreal, Quebec, Canada*, pages 85–91, ACM, New YOrk.

Senay, H., and Ignatius, E. (1994). A knowledge-based system for visualization design. *IEEE Computer Graphics and Applications*, 14(6):36–47.

Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley.

Stanton, A., and Unkrich, L. (Writers) (2003). Finding Nemo. In W. D. P. P. A. Studios (Producer), USA. Buena Vista Pictures / Walt Disney Pictures.

SWCP. (2003). Historical Data for S&P 500 Stocks. Retrieved October, 2006, from http://kumo.swcp.com/stocks

Tonnesen, D. (2001). Particle systems for artistic expression. In *Proceedings of Subtle Technologies Conference, Toronto, Canada*, pages 17–20, University of Toronto, Toronto.

Torgerson, W. S. (1952). Multidimensional scaling. *Psychometrika*, 17:401–419.

Tory, M., and Möller, T. (2004). Rethinking visualization: A high-level taxonomy. In *Proceedings of IEEE Symposium on Information Visualization (Infovis'04), Austin, Texas*, pages 151–158.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Press, Cheshire.

Upson, C., Faulhaber, T. A., Jr., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J. (1989). The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9:30–42.

van der Burg, J. (2000). Building an advanced particle system. *Game Developer*, 44–50.

Vande Moere, A. (2004). Time-varying data visualization using information flocking boids. In *Proceedings of Symposium on Information Visualization (Infovis'04), Austin, USA*, pages 97–104.

Vande Moere, A., and Clayden, J. J. (2005). Cellular ants: Combining ant-based clustering with cellular automata. In *Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pages 177-184.

Vande Moere, A., Mieusset, K. H., and Gross, M. (2004). Visualizing abstract information using motion properties of data-driven infoticles. In *Proceedings of Conference on Visualization and Data Analysis 2004 (IS&T/SPIE Symposium on Electronic Imaging)*, pages 33–44. San Jose, CA.

Vande Moere, A., Clayden, J. J., and Dong, A. (2006). Data clustering and visualization using cellular automata ants. In *Proceedings of ACS Australian Joint Conference on Artificial Intelligence (AI'06), Hobart, Australia*, pages 826–836, Springer, Berlin.

Von Neumann, J. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, Chicago.

Ware, C. (2000). *Information Visualization Perception for Design*. Morgan Kaufmann, San Francisco.

Ware, C., Neufeld, E., and Bartram, L. (1999). Visualizing causal relations. In *Proceedings of IEEE Symposium on Information Visualization (Infovis'99), San Francisco, CA*, pages 39–42, IEEE Computer Society, Washington, DC, USA.

Wojciech, B. (2001). Multivariate visualization techniques. Retrieved June 2006, from http://www.pavis.org/essay/multivariate_visualization_techniques.html

Woolridge, M. (2001). *Introduction to Multiagent Systems*. Wiley, New York.

Zambonelli, F., Mamei, M., and Roli, A. (2002). What can cellular automata tell us about the behavior of large multi-agent systems? In A. Omicini and J. Castro, editors, volume 2603 of *Lecture Notes in Computer Science*, pages 216–231. Springer, Berlin.

14

Emergence of Traveling Localizations in Mutualistic-Excitation Media

Andrew Adamatzky

14.1 Introduction

A localization in the context of nonlinear spatially extended systems means a localized, compact, and relatively long-living perturbation or a deviation from the resting state. Localizations are everywhere, e.g., breathers in molecular chains, excitons in molecular aggregates, dipole defects in tubulin microtubules, oscillons in granular materials, quasi-particles, and dissipative solitons in reaction-diffusion systems.

In a DNA molecule there are intrinsic local modes of nucleotide base displacement, which emerge due to localization of thermal fluctuations (Forinash et al. 1994). DNA-based self-localizations, or breathers, are mobile: when excited, a breather travels along DNA strands.

Quasi two-dimensional localizations, excitons, can be found in light-sensitive molecular arrays, a localized electronic excitation emerging when an electronic system is optically excited (Toyozawa 1983). This happens because of a competition between delocalization and localization: the delocalization relies on molecular resonance, which in turn is responsible for the transfer of excitation along the molecular array. A feature of localization is that it is based on an imbalance of forces activated by electronic charge redistribution. This acts on relevant atoms to find new equilibria of molecular configuration coordinates (Toyozawa 1983), but destroys the resonance of electronic excitation to neighboring sites.

Tubulin microtubules are one more example of molecular arrays supporting traveling localizations. A microtubule is an array of tubulin dimers rolled into a tube, where the dimers have electric dipoles. The array can be thought of as an array of dipole moments interacting with their closest neighbors by dipole-dipole forces (Brown and Tuszynski 1999).

Light bullets are three-dimensional localizations—three-dimensional solutions of Schrödinger's equation (Edmundson and Enns 1993). Results of numerical simulations of three-dimensional generalized nonlinear Schrödinger equations in models with alternative refractive indices demonstrate the propagation of stable light bullets with soliton-like behavior (Edmundson and Enns 1993).

Liquid crystals exhibit two-dimensional mobile localized states, called worms (Dennin et al. 1996). These worm states are localized in the direction that is perpendicular to the direction of the liquid crystal; they extend in parallel directions, with the convective amplitude being large at one end of the worm, called the head, and decaying gradually toward the other end (Riecke and Granzow 1998), while the worms propagate head forward.

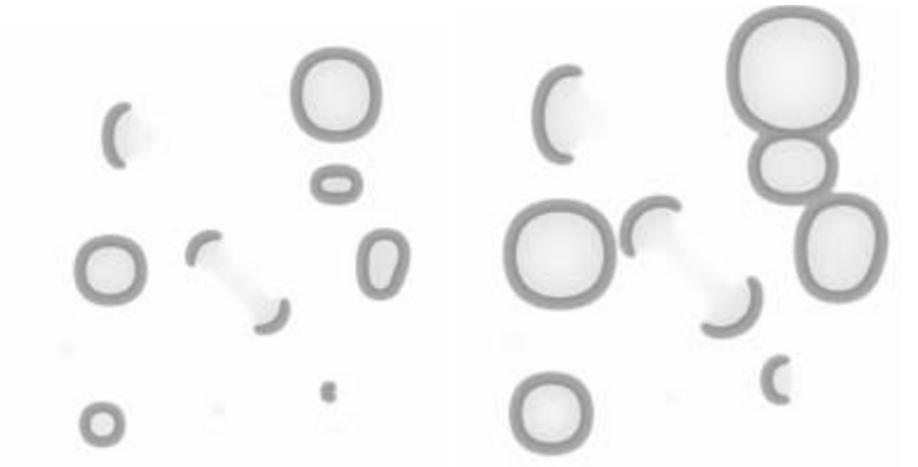
Localized wave fragments are typical for Belousov-Zhabotinsky media in a subexcitable mode. In this particular mode, a perturbation of the medium's characteristics causes the generation of 'classical' target or spiral waves, where compact wave fragments propagate for a reasonably long time without changing their shape (Sendiña-Nadal et al. 2001).

The excitable localizations can be simulated in two-variable Oregonator equations (Field and Noyes 1974) adapted to a light-sensitive reaction with applied illumination (Beato and Engel 2003):

$$\frac{\partial u}{\partial t} = \frac{1}{\epsilon}(u - u^2 - (fv + \phi)\frac{u - q}{u + q}) + D_u \nabla^2 u$$

$$\frac{\partial v}{\partial t} = u - v,$$

where the variables u and v represent local concentrations of bromous acid, $HBrO_2$, and the oxidized form of the catalyst ruthenium, Ru(III); ϵ sets up a ratio of the time scale of variables u and v; q is a scaling parameter depending on reaction rates; f is a stoichiometric coefficient; and ϕ is a light-induced bromide production rate proportional to intensity of illumination. The intensity of illumination is an excitability



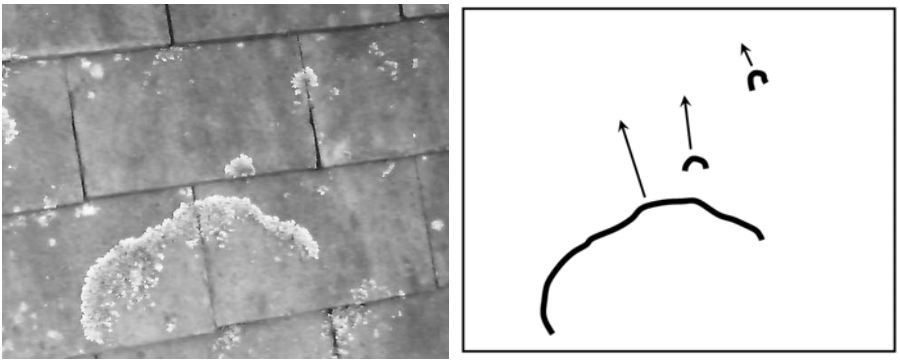
(a) (b)

Fig. 14.1. Two snapshots of a Belousov-Zhabotinsky medium in subexcitable mode. We can observe classical circular waves as well as localized wave fragments.

parameter, i.e., moderate light intensity will facilitate the excitation process, while higher intensity will produce excessive quantities of bromide, which suppresses the reaction. We assumed that the catalyst is immobilized in a thin layer of gel; therefore there is no diffusion term for v. To integrate the system we used Euler's method with a five-node Laplacian operator, time step $\triangle\ t = 10^{-3}$, and grid point spacing $\triangle\ x = 0.15$, with the following parameters: $\phi = \phi_0 + A/2$, $A = 0.0011109$, $\phi_0 = 0.0766$, $\epsilon = 0.03$, $f = 1.4$, $q = 0.002$. The chosen parameters correspond to a region of "higher excitability of the subexcitability regime" outlined in (Sendiña-Nadal et al. 2001) that supports propagation of sustained wave fragments (Fig. 14.1). This was one of the first ever localizations observed in two-dimensional macroscopic systems.

We can envisage that traveling localizations observed in subexcitable Belousov-Zhabotinsky medium are common features for all excitable systems in a subexcitable mode. For example, the localized wave fragments can be observed in lichen colonies (Fig. 14.2). However, are they common for ecological systems? Could such localizations be found in computational models of population dynamics? To get an answer we turned to cellular-automaton models of excitable quasi-chemical systems (Adamatzky 2005).

In Section 14.2 we introduce a novel model of excitable cellular automata, where every resting cell becomes excited depending not only on the number of excited neighbors but also on the number of refractory neighbors. In excitable systems, excited states are always followed by refractory states; therefore, excited states can be seen as autotrophs and refractory states as heterotrophs. The discussed rule of excitation implies that the persistence of autotrophs relies on the existence of heterotrophs. Basic modes of excitation dynamics, or an excitable medium's responses to random stimulation, are classified in Section 14.3. Minimal mobile localizations are discussed in



(a) (b)

Fig. 14.2. Traveling wave fragments observed in colonies of lichens: (a) photograph, (b) scheme of the wave fronts and their supposed velocity vectors.

Section 14.4, and a few examples of heavyweight localizations are provided in Sect. 14.5. Section 14.6 outlines domains of cell-state transition rule space that support the richest localization dynamics. The potential impact of the findings on cellular-automaton theory and nonlinear sciences is discussed in Sect. 14.8.

14.2 Mutualistic Excitation Rules for Cellular-Automaton Dynamics

Cellular automata have proved to be computationally efficient, accurate, and user-friendly tools for simulation of huge varieties of spatially extended systems (Ilachinski 2001; Chopard and Droz 2005). They are essential in computational analyses of nonlinear systems and in the exhaustive search for nontrivial functions in cell-state transition rule spaces (Adamatzky 2001; Adamatzky et al. 2006). Cellular-automaton representation of reaction-diffusion and excitable systems is of particular importance because it allows us to effortlessly map already established massively parallel architectures onto novel material bases of chemical systems and design nonclassical and nature-inspired computing architectures (Adamatzky et al. 2005).

Based on our expansion of the classical Greenberg-Hasting model (Greenberg and Hastings 1978) to interval excitation rules (Adamatzky 2001) and productive adoption of interval rules in modeling reaction-diffusion systems with binary-state automata (Adamatzky et al. 2006), we designed a unique model of an excitable cellular automaton with interval-based rules of mutualistic excitation. In this automaton every resting cell becomes excited depending not only on the number of excited neighbors but also on the number of refractory neighbors. We discuss the results of our studies of over a thousand rules of interval mutualistic excitation and provide classification and characterization of basic mobile localizations, or gliders as they are usually referred to in cellular automata theory.

Every cell of the automaton takes three states: resting ($\cdot$), excited ($\bullet$), and refractory ($\circ$). Cells update their states in parallel by the same rule, depending on the states of their immediate neighbors. A cell x 'calculates' the number of excited, $\sigma_+^t(x)$, and refractory, $\sigma_-^t(x)$, neighbors:

$$\sigma_a^t(x) = \sum_{y \in u(x)} \{y : y = a\}, \text{ where } u(x) = \{y : |x - y| <= 1\}.$$

A resting cell becomes excited if the number of excited neighbors belongs to interval $[\theta_1, \theta_2]$ and the number of refractory neighbors belongs to interval $[\delta_1, \delta_2]$. An excited cell becomes refractory and a refractory cell goes to resting stage unconditionally, independent of the states of their neighbors. The cell state-transition rule is as follows:

$$x^{t+1} = \begin{cases} \bullet, \text{ if } x^t = \cdot \text{ and } \sigma_+^t(x) \in [\theta_1, \theta_2] \text{ and } \sigma_-^t(x) \in [\delta_1, \delta_2] \\ \circ, \text{ if } x^t = \bullet \\ \cdot, \text{ otherwise} \end{cases} .$$

Henceforth, rules are denoted by tuples $R(\theta_1\theta_2\delta_1\delta_2)$.

We call the excitation mode described by the functions above *mutualistic* because, for a cell to be excited requires not only excited neighbors but also neighbors in refractory states. If we interpret the excited and refractory modes as species, then, for each species to exist (or in the case of excitation state, to spread to new territories), another species must be present.

Our previous experience (Adamatzky et al. 2005; Wuensche and Adamatzky 2006) shows that interpretation in terms of chemical systems can be quite productive, and, moreover, may lead to possible chemical laboratory implementations of the discovered phenomena (Adamatzky et al. 2005).

The mutualistic excitation rule can be written as the following set of quasi-chemical equations:

$$\begin{aligned} S + aE + bI &\rightarrow E \\ E &\rightarrow I \\ I &\rightarrow S, \end{aligned}$$

where S is a substrate (resting cell state), E is an active form of some reagent (excitation cell state), and I is an inactivated form of the reagent (refractory cell state). Reaction coefficients a and b are derived from the exact structure of the rule $R(\theta_1\theta_2\delta_1\delta_2)$. We could not establish an immediate direct correspondence between our abstract model and any real system. One could say that the Belousov-Zhabotinsky reaction would be the closest match in terms of the phenomenology of excitation: in subexcitable media traveling wave-fragments, or quasi-particles, are induced by random configuration of illumination—light noise (Wang et al. 1999). BZ-AOT modification (Vanag and Epstein 2001) of the Belousov-Zhabotinsky reaction is very promising because it allows for the coexistence of classical spot, stripe, and labyrinthine patterns. A simple surface oxidation system exhibiting mobile localizations (Rotermund et al. 1991) also shows excellent possibilities for achieving experimental implementation of the studied systems with a mutualistic excitation mode.

In terms of 'very abstract' ecological systems, we can say that I is a heterotroph, which consumes E. Species E are autotrophs, which live on the substrate S. The key point is that for autotroph E to propagate over new territories, the presence of heterotroph I is essential.

14.3 Phenomenology

We have tested the behavior of each rule by 'stimulating' an automaton lattice at random. On the initially resting lattice we selected discs of cells centered in the lattice midpoint, and assigned one of three states with probability $\frac{1}{3}$ to each cell of the disc. Among over a thousand rules studied, just one-third exhibited excitation dynamics persisting for at least a few steps of lattice development. Several classes of rules, based on spatiotemporal dynamics of excitation, were selected.

So-called homogeneous dynamics is most frequently found among the studied automata. Excitation in the originally randomly excited disc remains quasi-chaotic

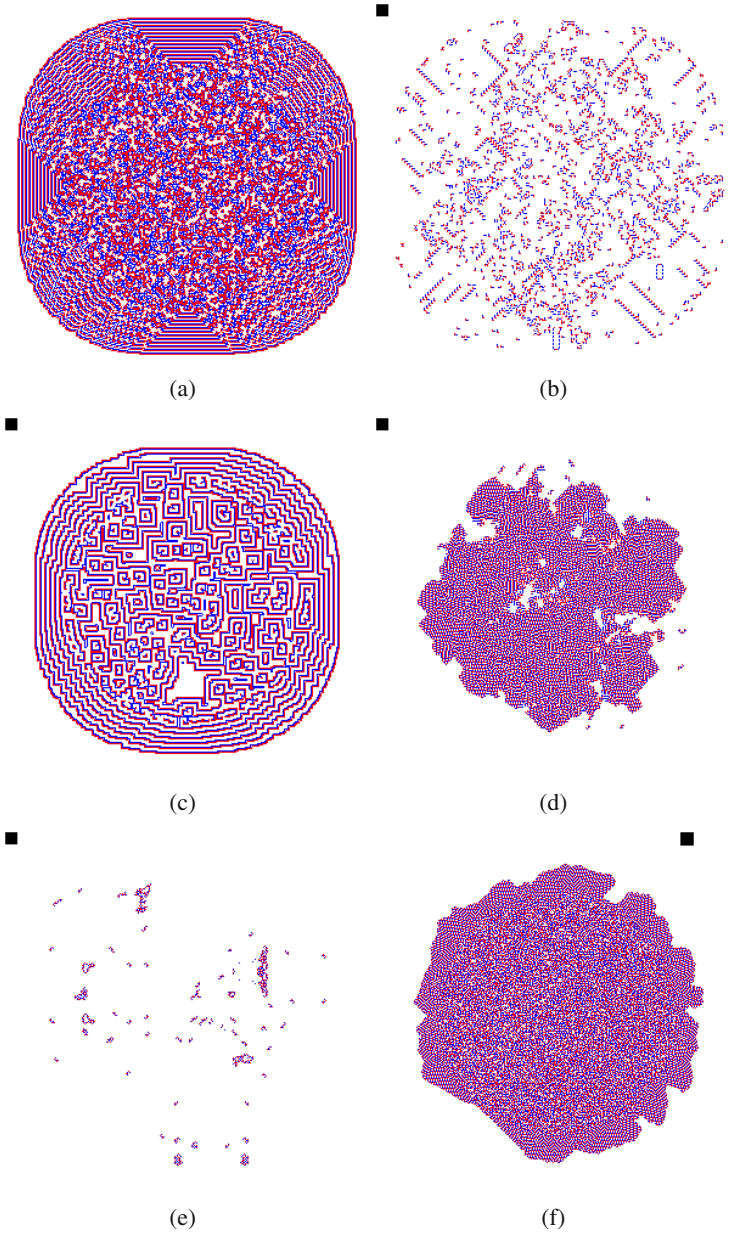


Fig. 14.3. Snapshots of exemplar configurations generated by functions (a) $R(1306)$, (b) $R(1117)$, (c) $R(1401)$, (d) $R(1428)$, (e) $R(2413)$, and (f) $R(2518)$. Cellular automaton 300×300 cells, initially each cell less 100 cells from center of the lattice took one of the states at random.

without any accentuated features. When the excitation pattern spreads, excitations at the growing fronts merge, forming a dense hierarchy of spreading wave fronts (Fig. 14.3a).

Automata whose dynamics are very rich with or gliders, are grouped in the second class. Single gliders are easily detectable on the peripheral parts of the original randomly excited domain. However, in the central part of the domain, collisions between gliders produce more glides, and the configuration starts to look more like a chaotic excitation (Fig. 14.3b).

A domain of a 'classical' spiral waves encircled by several fronts of target waves is formed after initial stimulation of the automata belonging to the third class of rules. The target waves are in sites where excited cells are adjacent to refractory cells. Wave fronts of the target waves merge at the periphery and then propagate as united fronts of target waves (Fig. 14.3c).

Rules of the forth class produce remarkable, very slowly expanding domains of the labyrinth-like configurations (Fig. 14.3d). The patterns even keep their shape; however it will be initially arbitrary for a long time. The wave fronts there do not obey the curvature-speed principle, and thus look more like morphological patterns in living systems than simple excitable systems. Gliders are seen on the periphery of the spreading pattern; they move faster than the growing pattern itself.

The fifth class gives us configurations where only mobile localizations and generators of mobile localizations (glider guns) exist (Fig. 14.3e).

A slowly growing blob, with a quasi-chaotic homogeneous core and a labyrinth-like patterned shell, is typical for the rules of sixth class (Fig. 14.3f).

14.4 Mobile Localizations

When classifying localizations, we introduce the term weight. A localization's weight is the number of cells occupied by the localization's nonresting states (excitation and refractory states).

14.4.1 Minimal Gliders: Weight 4

The minimal mobile localization E_2^2 that we found consists of two excitation states (excitation head) and two refractory states (refractory tail); the localization travels excited head forward (Fig. 14.4a). This is the only localization found in rule $R(2200)$; it is also present but not unique in $R(2201)$. Localization E_2^2 and its variations with three excited and three refractory states are the only gliders in rules $R(2300)$, $R(2400)$, $R(2500)$, $R(2600)$, $R(2700)$, and $R(2800)$. Varieties of the localization E_2^2—extended wave fragments—are shown in Fig. 14.4b–d; they are typical for rules $R(2201)$, $R(2301)$, $R(2401)$, $R(2501)$, $R(2601)$, $R(2701)$, and $R(2801)$. In rules $R(2202)$, $R(2302)$, $R(2402)$, $R(2502)$, $R(2602)$, $R(2702)$, and $R(2802)$ branching localizations are most common, and the length of lateral branches can vary (Fig. 14.4e).

The glider E_2^2 has a weight of four states, and a speed equivalent to the 'speed of light,' i.e., it moves along horizontal and vertical rows of the cellular-automaton

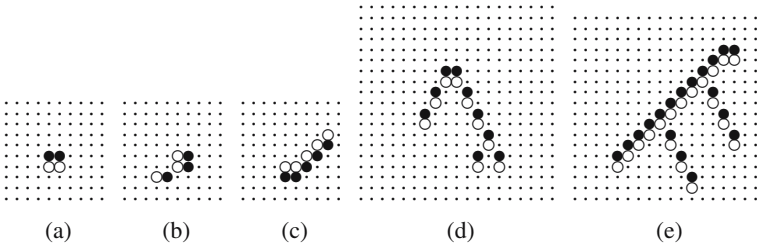


Fig. 14.4. Mobile localizations: (a) E_2^2 and those derived from E_2^2 traveling north; (b) shortest extended localization traveling east; (c) typical extended localizations traveling south; (d) branching extended localization traveling north; (e) multiply branched traveling north.

array. There is another minimal mobile localization with two excited and two refractory states, the glider ${}_D E_2^2$, which travels along diagonals of the lattice with speed of $\frac{1}{2}$ cells per time unit (Fig. 14.5). The glider is typical for rules $R(1112)$, $R(1113)$, $R(1114)$, $R(1115)$, $R(1116)$, $R(1117)$, and $R(1118)$.

14.4.2 Gliders with Weight 6

Among gliders with constant weight 6, G_{5a}^1 and G_{5b}^1, (Fig. 14.6), period two, have the most 'unbalanced' ratio of excited to refractory states. At one step they have five excited and one refractory state, and at the next step one excited and five refractory states. They change their global state with period two. These gliders exist in rules $R1215$, $R(1216)$, $R(1217)$, and $R(1218)$.

Glider G_3^3, period four, always has three excited and three refractory states, as shown in Fig. 14.6c. This is the only glider ever observed in rules $R(2213)$, $R(2215)$, $R(2216)$, $R(2217)$, and $R(2218)$; and is one of many mobile localizations observed in rules $R(2613)$, $R(2813)$, and $R(2313)$. Several gliders G_3^3 may couple together to form extended wave fragments, as shown in Fig. 14.7.

Glider E_3^3 is the slowest glider among those with weight 6: it changes its configuration over a period of eight steps (Fig. 14.6d,e). This glider is very rare but can be found in the development of initial random configurations in rules $R(1422) \ldots R(1428)$, and rules $R(1522)$, $R(1622)$, $R(1722)$, $R(1822)$, $R(2214)$, and $R(2413)$.

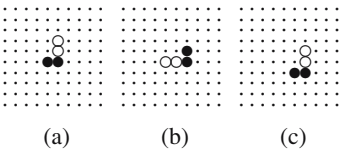


Fig. 14.5. Glider ${}_D E_2^2$ moving southeast.

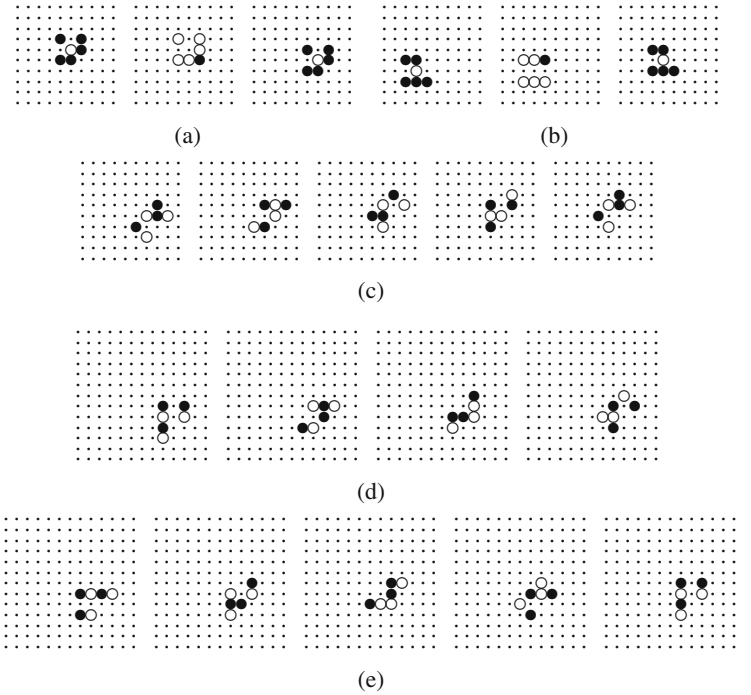


Fig. 14.6. Gliders with constant weight 6: (a) glider G^1_{5a} traveling southeast; (b) glider G^1_{5b} traveling northeast; (c) glider G^3_3 traveling southwest; and (d,e) glider E^3_3 traveling northwest.

14.4.3 Glider with Weight Changing between 6 and 8

The glider G_{68} alters its configuration in four time steps, during which its weight varies from 6 to 8 and back again (Fig. 14.8). These mobile localizations are typically found in configurations generated by rules $R(1112), R(1113), \ldots, R(1118)$.

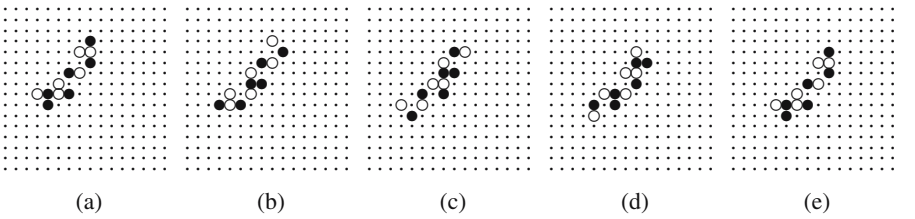
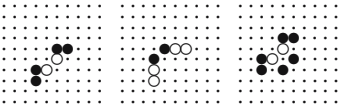


Fig. 14.7. Two gliders G^3_3 are associated together as a single wave fragment.



(a)

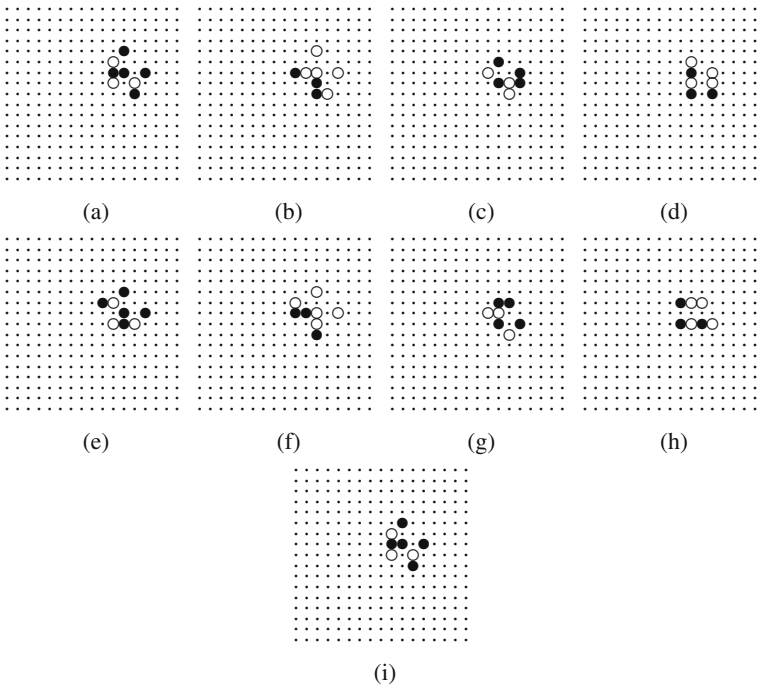
Fig. 14.8. Glider G_{68} traveling northwest.

14.4.4 Glider with Weight 7 or 8

The cellular automaton governed by rule $R(1323)$ exhibits glider G_{78} (Fig. 14.9), period eight, which updates its weight in the following cyclic pattern $8 \to 8 \to 7 \to 7 \to 8 \to 8 \to 8 \to 7 \to 8$.

14.4.5 Gliders with Constant Weight 8

Rule $R(1214)$ supports glider G_8 with weight eight (Fig. 14.10a), period two. A second glider with weight 8, E_8, period four, observed in configurations of the cellular automaton is governed by rule $R(2313)$ (Fig. 14.10b).



(a) (b) (c) (d)

(e) (f) (g) (h)

(i)

Fig. 14.9. Glider G_{78} traveling southwest.

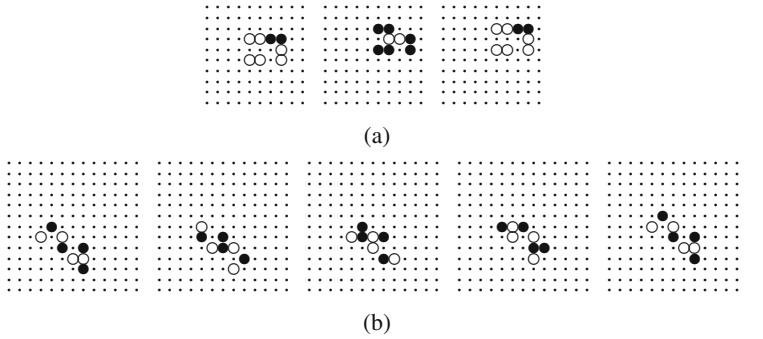


Fig. 14.10. Gliders with weight 8: (a) glider G_8 traveling southeast; (b) glider E_8 traveling northeast.

14.4.6 Gliders with Weight Switching between 8 and 10

Glider $G_{8/10}$ (Fig. 14.11), period four, changes its weight in the sequence: $10 \rightarrow 8 \rightarrow 8 \rightarrow 10 \rightarrow 10$, corresponding to variations in the number of excited states are $4 \rightarrow 4 \rightarrow 4 \rightarrow 6 \rightarrow 4$. The glider $G_{8/10}$ is a typical mobile localization for rules $R(2313)$, $R(2413)$, and $R(2513)$.

14.4.7 Gliders with Weight Switching between 9 and 10

Glider $G_{9/10}$ (Fig. 14.12), period eight, changes its weight in the following sequence $10 \rightarrow 9 \rightarrow 9 \rightarrow 10 \rightarrow 10 \rightarrow 9 \rightarrow 9 \rightarrow 10 \rightarrow 10$. Such localizations are usually found in configurations generated by rules $R(1523)$, $R(1623)$, and $R(1723)$.

14.4.8 Glider with Weight Switching between 9 and 13

Glider $G_{9/13}$ (Fig. 14.13), period eight, is observed in rules $R(2313)$ and $R(2413)$. Its average weight oscillates around 10 and reaches its maximum 13 just once in the configuration cycle: $9 \rightarrow 10 \rightarrow 10 \rightarrow 10 \rightarrow 12 \rightarrow 13 \rightarrow 11 \rightarrow 9 \rightarrow 9$.

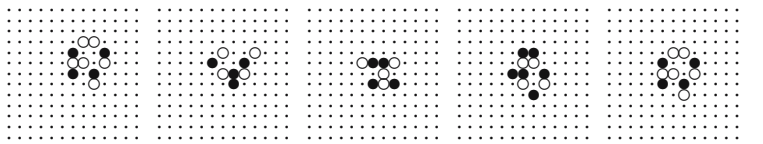


Fig. 14.11. Glider $G_{8/10}$ traveling southwest.

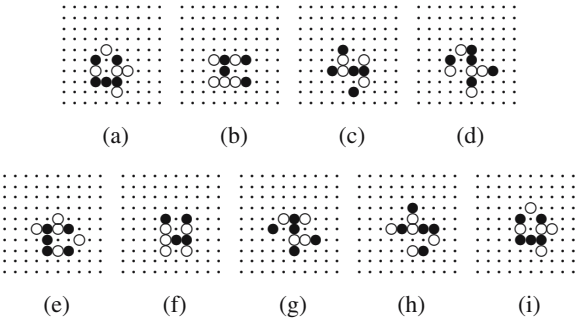


Fig. 14.12. Glider $G_{9/10}$ traveling northwest.

14.4.9 Glider with Weight Changing between 9 and 14

The glider $G_{9/14}$ has one of the longest periods. It repeats its configuration only once in 17 steps of evolution (Fig. 14.14), observed in rule $R(1423)$. The weight of the glider cycles as follows: $14 \to 14 \to 12 \to 12 \to 14 \to 12 \to 11 \to 11 \to 9 \to 10 \to 12 \to 12 \to 10 \to 11 \to 12 \to 12 \to 14$.

14.4.10 Glider with Constant Weight 11

The glider G_{11} has period two, and oscillates between configurations with nine excited and two refractory states and two excited and nine refractory states (Fig. 14.15). This glider can be found in the rules $R1315$, $R(1316)$, and $R(1317)$.

14.4.11 Glider with Constant Weight 12

The glider G_{12} with constant weight 12 is observed in rules $R(1316)$, $R(1317)$, and $R(1318)$. While the overall weight of the glider is preserved during its evolution, and

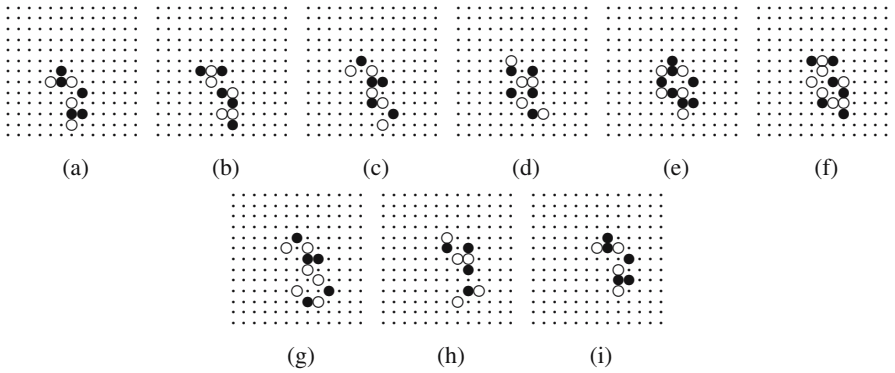


Fig. 14.13. Glider $G_{9/13}$ traveling northeast.

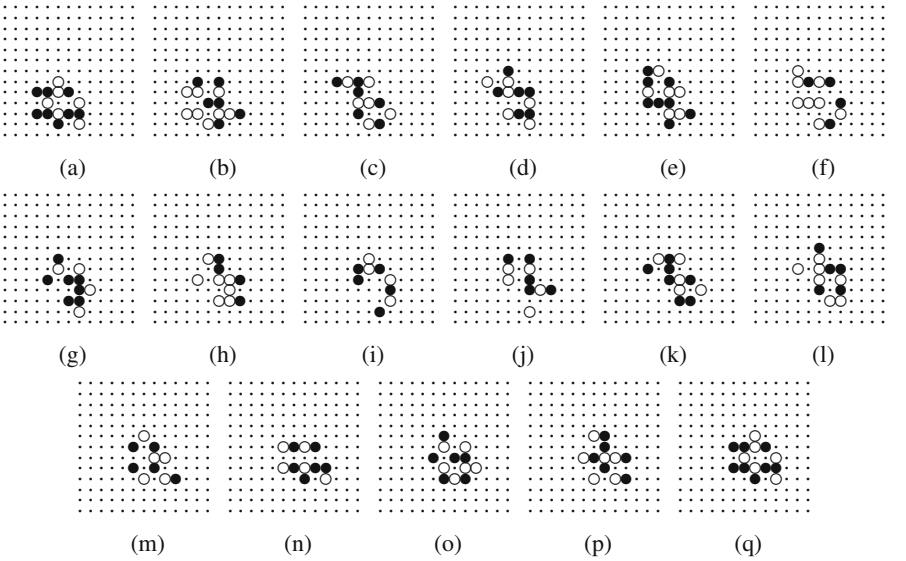


Fig. 14.14. Glider $G(9/14)$ traveling northeast.

the glider has period two, the number of excited states in the glider changes as follows: $3 \to 9 \to 3$ (Fig. 14.16).

14.4.12 Glider with Constant Weight 13

Glider G_{13}, as well as G_{19} is a 'relative' of glider G^1_{5a}. It has period two and preserves its weight, but the number of excited states in the glider configuration switches between three and ten (Fig. 14.17). The glider is typical for rule $R(1214)$.

14.4.13 Glider with Weight Switching between 13 and 14

Glider $G_{13/14}$ (Fig. 14.18) changes its weight between 13 and 14 during its life cycle of eight steps. The glider is found in rules $R(1523)$, $R(1623)$, and $R(1723)$. The glider's weight varies as follows: $13 \to 13 \to 14 \to 14 \to 13 \to 13 \to 14 \to 14 \to 13$.

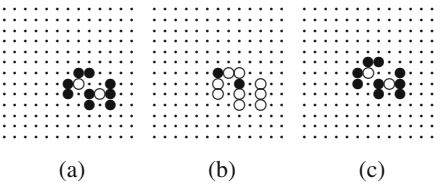


Fig. 14.15. Glider G_{11} traveling northwest.

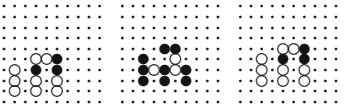


Fig. 14.16. Glider G_{12} traveling northeast.

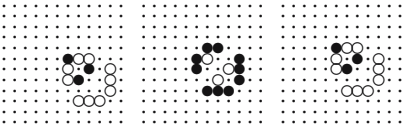


Fig. 14.17. Glider G_{13} traveling northwest.

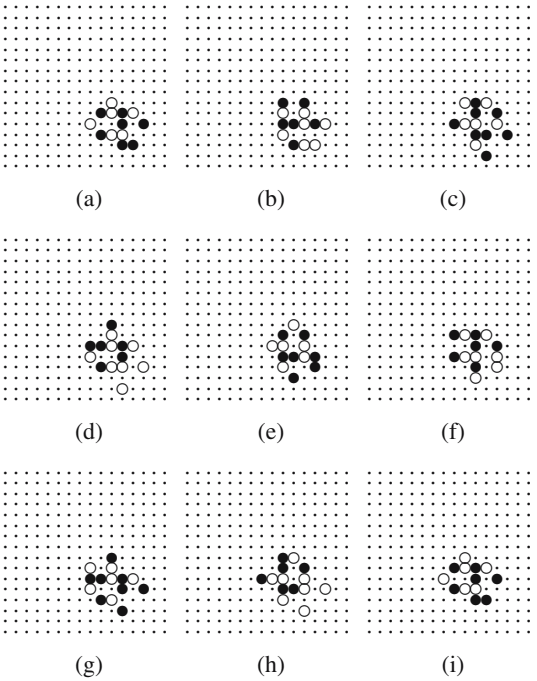


Fig. 14.18. Glider $G_{13/14}$ traveling northwest.

14.4.14 Glider with Constant Weight 15

Glider G_{15}, weight 15, period two, is one of several gliders traveling along rows or columns of the cellular-automaton lattice (Fig. 14.19). This is a glider typical for rules $R(2203)$, $R(2303)$, $R(2403)$, $R(2603)$, $R(2703)$, $R(2803)$, $R(2201$, and $R(2202)$. The glider is minimal for this type, but has many heavier variations with larger configurations of tails.

14.4.15 Glider with Weight Switching between 15 and 18

Glider $G_{15/18}$, found in rules $R(2813)$, $R(2713)$, and $R(2613)$, has period four, and its weight varies as follows: $18 \to 16 \to 15 \to 15 \to 18$ (Fig. 14.20).

14.4.16 Glider with Constant Weight 19

The glider G_{19} relates to G^1_{5a} and G_{13}. It has period four, preserves its weight, and switches in number of excited states between five and fourteen (Fig. 14.21). Cellular automata governed by rules $R(1213)$ and $R(1214)$ exhibit such gliders in their development.

14.4.17 Glider Changing Its Weight from 21 to 22

Glider $G_{20/22}$ (Fig. 14.22), period four, observed in rules $R(2314) \ldots R(2318)$, alters its weight gradually as $21 \to 21 \to 22 \to 21 \to 21$. Its excitation component switches between 10 and 11 every two steps of automaton evolution as follows: $11 \to 10 \to 11 \to 10 \to 11$.

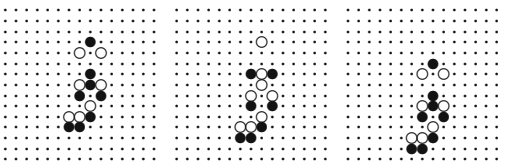


Fig. 14.19. Glider G_{15} traveling south.

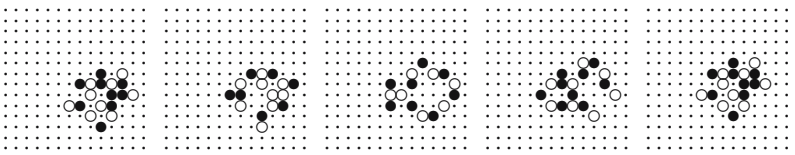


Fig. 14.20. Glider $G_{15/18}$ traveling northwest.

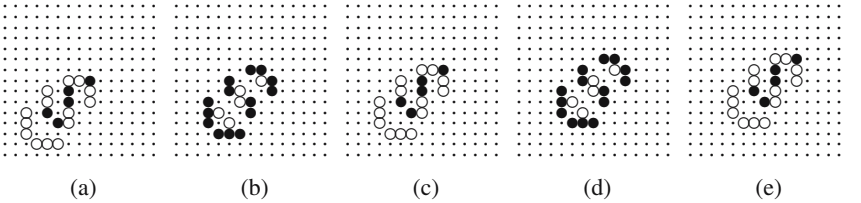


Fig. 14.21. Glider G_{19} traveling southeast.

14.5 Huge Mobile Localizations

Huge mobile localizations, with weights over 30, are not uncommon in the rules investigated. They are, however, very sensitive to perturbations. If care is not taken immediately after they have emerged, the localizations are usually destroyed in collisions with smaller localizations and gliders. An example of a very large mobile localization, the weight of which exceeds 43, is shown in Fig. 14.23. It has period eight and moves northeast.

The large traveling localization shown in Fig. 14.24 resembles propagating wave fragments in a subexcitable Belousov-Zhabotinsky system (Sendiña-Nadal et al. 2001; Adamatzky 2004), the wave fragment grows from the center (Fig. 14.24k,l) and the excitation spreads toward the ends of the fragment (Fig. 14.24l,m), where it dies (Fig. 14.24n).

An even more complicated wave fragment is shown in Fig. 14.25. The head of the pattern has a hollow structure, with excitation developing on the contour of the head, thus causing forward motion of the whole pattern.

14.6 Characterizing Localizations

The following characteristics were used to analyze the sets of localizations discovered. Weight is the number of nonresting cells in a localization. Imbalance is the absolute difference between excited and refractory states in a localization divided by the weight of the localization. Density is a localization's weight divided by the size of a rectangle occupied by the localization. Period is the number of steps a localization needs to restore its state.

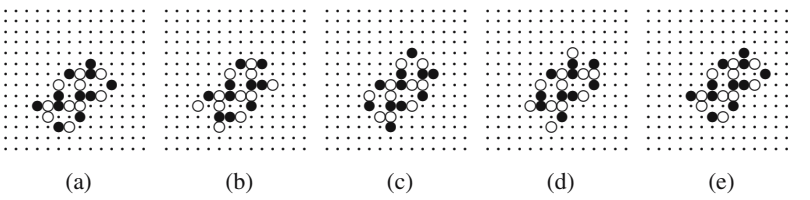


Fig. 14.22. Glider $G_{20/22}$ traveling northeast.

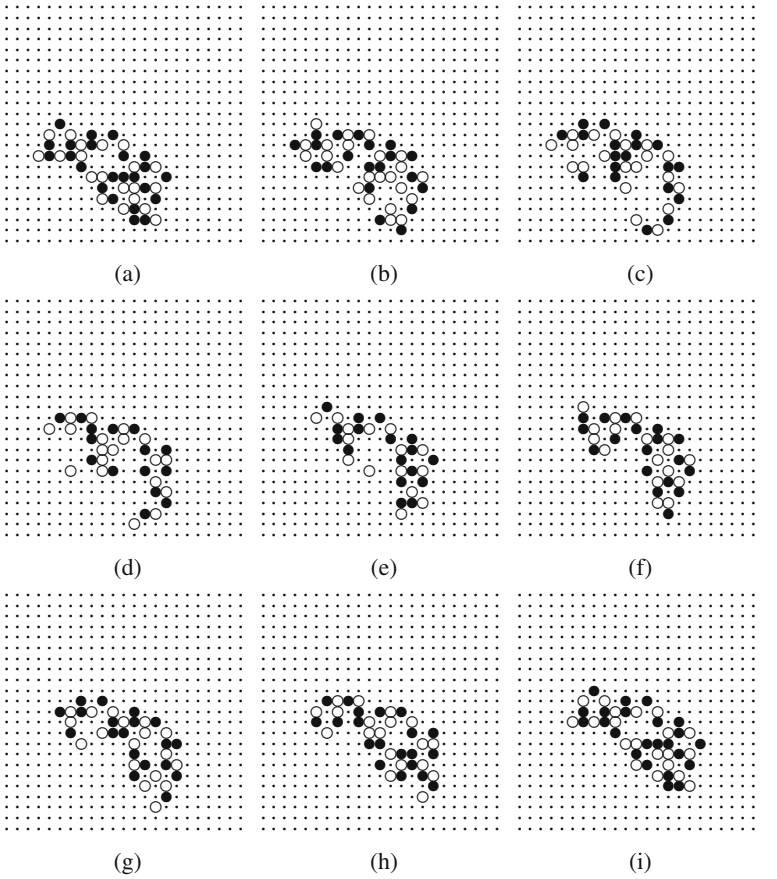


Fig. 14.23. Large mobile localization observed in rule $R(2713)$, traveling northeast.

Figure 14.26 shows that, overall, the period of gliders increases with an increase in the glider's weight till the weight reaches ten nonresting cells. Then the period starts to decrease. Localizations with maximum period, the longest time to restore their states, have weight around 10. Density, in general, decreases with the increase in a glider's weight. The more balanced the glider, the longer the period of its oscillation. Highly imbalanced gliders usually change their state in a small number of time steps.

14.7 Excitation Rules Rich with Localizations

In terms of the number of different species of localizations, what are the richest excitation rules? The answer is in the matrices of localizations' frequencies shown in Fig. 14.27. Each matrix $\mathbf{M} = (M_{ij})_{0 \le i,j \le 8}$ is constructed as follows: Initially, all entries of the matrix are nil's. Then for each localization found in configurations generated by rule $R(abcd)$: $M_{ij} \leftarrow M_{ij} + 1$ if $i \in [\theta_1, \theta_2]$ and $j \in [\delta_1, \delta_2]$. When all

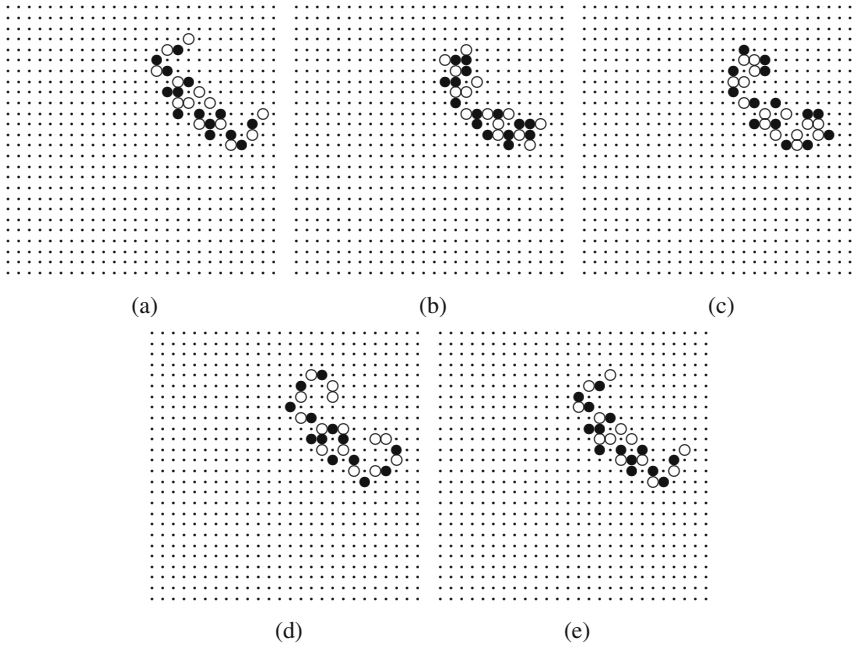


Fig. 14.24. Large mobile localization discovered in rule $R(2813)$; it also exists in rules $R(2613)$ and $R(2713)$.

entries are updated, each entry M_{ij} shows a number of species of mobile localizations supported by a rule: a resting cell becomes excited if it has i excited neighbors and j refractory neighbors.

From the matrices in Fig. 14.27 we see that rule $R(2222)$ is the richest in localizations, supporting 11 species of gliders. The automata governed by the excitation intervals, which include rule $R(2222)$, are extended versions of the 2^+ medium, studied in detail previously (see, e.g., Adamatzky 2001).

For mobile localizations, their 'richness' has its maximum in $R(2323)$ (10 and 11 species), and then gradually decreases, as if forming a diffusive profile (Fig. 14.27a), a bit biased toward the direction of increasing indices of columns and rows of the matrix.

14.8 Discussion

We have studied spatiotemporal behavior of cellular-automaton models of excitable systems, where the excitation of a cell depends not only on the number of excited neighbors, as in classical models of excitation, but also on the number of refractory neighbors. Namely, every resting cell becomes excited if the number of excited neighbors is between θ_1 and θ_2 and the number of refractory neighbors is between δ_1 and

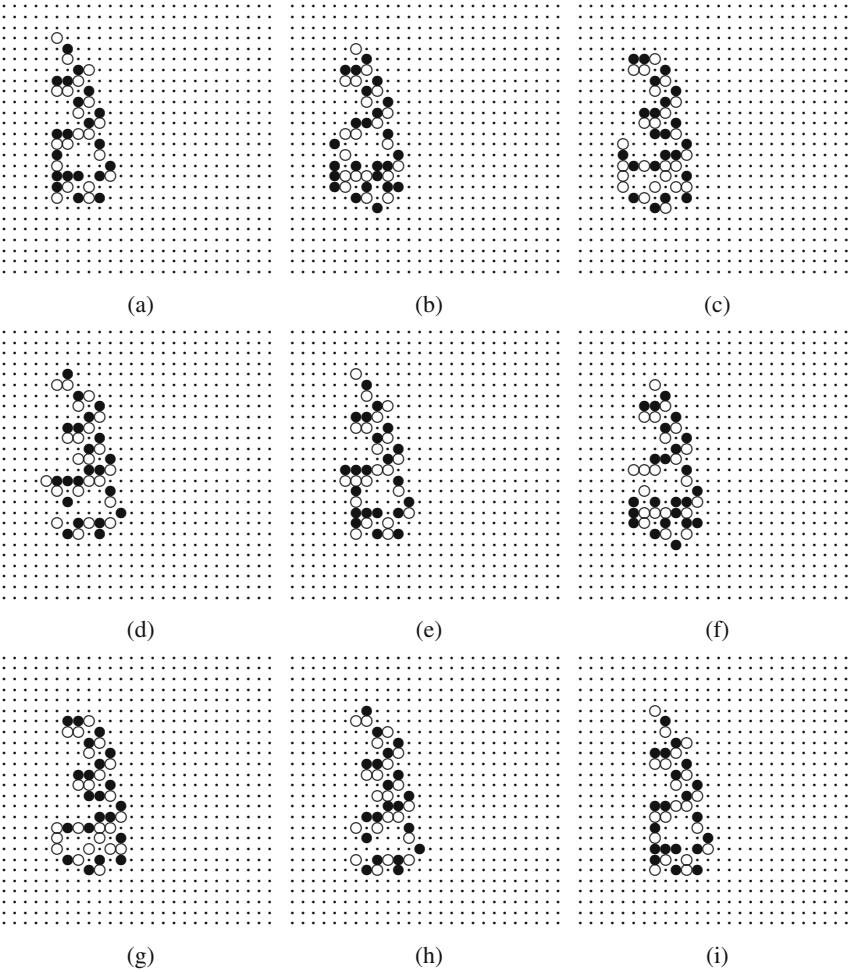


Fig. 14.25. Complex traveling pattern observed in rule $R(2813)$; it can also be found in rules $R(2814)$ and $R(2818)$.

δ_2. Not all combinations of excitation intervals are valid, but only intervals where $0 \leq \theta_1 \leq \theta_2 \leq 8$, $0 \leq \delta_1 \leq \delta_2 \leq 8$, and $\theta_1 + \theta_2 + \delta_1 + \delta_2 \leq 8$.

Such rules can be interpreted in terms of biological ecosystems as mutualistic, as both excited and refractory species derive benefits from their interactions, or at least coexist in the same local domain, or cell neighborhood. Refractory species 'benefit' because every excited species becomes refractory. Excited species 'benefit' owing to the nature of the excitation rule defined. Rules with rich localization dynamics have nonzero δ_1, meaning that mutualism is obligatory for both excited and refractory species.

Glider	Weight	Imbalance	Density	Period
E_2^2	4	0	1	1
${}_D E_2^2$	4	0	2/3	2
G_{5a}^1	6	2/3	2/3	2
G_{5b}^1	6	2/3	1/2	4
G_3^3	6	0	3/8	3
E_3^3	6	0	3/8–1/2	4
G_{68}	6–8	1/3–1/2	3/8–1/2	3
G_{78}	7–8	1/7–1/4	7/16–2/5	8
G_8	8	1/2	2/3	2
E_8	8	0	3/10–8/25	4
$G_{8/10}$	8–10	0–1/5	1/3–1/2	4
$G_{9/10}$	9–10	0–1/9	1/2–3/4	8
$G_{9/13}$	9–13	0–1/9	13/24–3/8	8
$G_{9/14}$	9–14	0–1/7	2/5–14/25	16
G_{11}	11	7/11	11/20	2
G_{12}	12	3/4	3/5	2
G_{13}	13	10/13	13/25	2
$G_{13/14}$	13–14	0–1/7	7/18–13/30	8
G_{15}	15	1/15	3/8–5/12	2
$G_{15/18}$	15–18	0–1/15	5/14–3/7	4
G_{19}	19	14/19	19/49	4
$G_{20/22}$	20–22	0–1/21	3/8–11/28	4

Fig. 14.26. Characteristics of mobile localizations or gliders.

While referring to ecological models, we should emphasize that the architecture of traveling localizations discovered in our computational experiments is different from that of autowaves typical for prey-predator population systems. In prey-predator systems, the wave fronts are usually represented by autotrophs and the wave tails by heterotrophs. Mutualistic interaction between species in an ecological system is not so straightforward; therefore every localization is an inseparable combination of auto- and heterotrophs merged and woven into integral propagating patterns.

The discovered structures make also a substantial contribution to the body of the "science of gliders"—the basics of the artificial life discipline—which is based on the original mobile localizations in Conway's Game of Life (see Gardner 1970) and its modifications such as Life 1133 (Heudin 1996) and Larger than Life (Griffeath and Moore 2003), as well as gliders in reaction-diffusion cellular automata (Adamatzky 2001; Wuensche 2004; Adamatzky et al. 2006).

$[\theta_1, \theta_2]$ \ $[\delta_1, \delta_2]$	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	3	7	8	4	4	3	3	2
2	2	7	11	11	5	5	4	4	3
3	2	6	10	10	3	3	2	2	1
4	2	4	6	6	0	0	0	0	0
5	2	3	4	4	0	0	0	0	0
6	2	2	4	4	0	0	0	0	0
7	2	2	4	4	0	0	0	0	0
8	2	2	2	2	0	0	0	0	0

Fig. 14.27. Localization frequency matrices for mobile localizations, or gliders.

References

Adamatzky, A. (2001). *Computing in Nonlinear Media and Automata Collectives*. IOP, Bristol and Philadelphig.

Adamatzky, A. (2004). Collision-based computing in Belousov-Zhabotinsky medium. *Chaos, Solitons & Fractals*, 21:1259–1264.

Adamatzky, A. (2007). Localizations in cellular automata with mutualistic excitation rules. *Chaos, Solitons & Fractals* in press.

Adamatzky, A., De Lacy Costello, B., and Asai T. (2005). *Reaction-Diffusion Computers*. Elsevier.

Adamatzky, A., Martínez, G. and Juan, C. Seck Tuoh Mora (2006). Phenomenology of reaction-diffusion binary-state cellular automata. *Int. J. Bifurcation and Chaos*, in press.

Alonso-Sanz, R., and Martin, M. (2006). Elementary cellular automata with elementary memory rules in cells: The case of linear rules. *Journal of Cellular Automata*, 1:70–86.

Aubry, S. (1997). Breathers in nonlinear lattices: Existence, linear stability and quantization. *Physica D*, 103:201–250.

Beato, V., and Engel, H. (2003). Pulse propagation in a model for the photosensitive Belousov-Zhabotinsky reaction with external noise. In Schimansky-Geier, L., Abbott, D., Neiman, A., Van den Broeck, C., editors, *Noise in Complex Systems and Stochastic Dynamics*, volume 5114 of *Proceedings of SPIE*, pages 353–362, SPIE.

Bode, M., Liehr, A. W., Schenk, C. P., and Purwins, H.-G. (2002). Interaction of dissipative solitons: Particle-Like behaviour of localized structures in a three-component reaction-diffusion system. *Physica D*, 161: 45–66.

Brown, J. A. and Tuszynski, J. A. (1999). A review of the ferroelectric model of microtubules. *Ferroelectrics*, 220:141–156.

Chopard, B. and Droz, M. (2005). *Cellular Automata Modeling of Physical Systems*, Cambridge University Press.

Dennin, M., Treiber, M., Kramer, L., Ahlers, G., and Cannell, D. S. (1996). Origin of traveling rolls in electroconvection of nematic liquid crystals. *Phys. Rev. Lett.*, 76:319–322.

Edmundson, D.E. and Enns, R.H. (1993). Fully 3–dimensional collisions of bistable light bullets. *Optics Letters*, 18:1609–1611.

Field, R. J., and Noyes, R. M. (1974). Oscillations in chemical systems. IV. Limit cycle behavior in a model of a real chemical reaction. *Journal of Chemical Physics*, 60:1877–1884.

Forinash, K., Peyrard, M., and Malomed, B. (1994). Interaction of discrete breathers with impurity modes. *Physical Review E*, 49:3400–3411.

Greenberg, J. M., and Hastings, S. P. (1978). Spatial patterns for discrete models of diffusion in excitable media. *SIAM Journal of Applied Mathematics*, 34:515–523.

Ilachinski, A. (2001). *Cellular Automata: A Discrete University*. World Scientific, Singapore.

Heudin, J.-K. (1996). A new candidate rule for the game of two-dimensional life. *Complex Systems*, 10:367–381.

Gardner, M. (1970). Mathematical Games—The fantastic combinations of John H. Conway's new solitaire game Life. *Scientific American*, 223:120–123.

Griffeath, D., and Moore, C. (2003). *New Constructions in Cellular Automata*, Oxford University Press.

Martínez, G. J., Adamatzky, A., and McIntosh, H. (2006). Localization dynamic in binary two-dimensional cellular automaton: Diffusion Rule. *Journal of Cellular Automata*, in press .

Maruno, K. and Biondini, G. (2004). Resonance and web structure in discrete soliton systems: the two-dimensional Toda lattice and its fully discrete and ultra-discrete analogues. *Journal of Physics A: Mathematical and General*, 37:11819–11839.

Riecke, H., and Granzow, G.D. Localization of waves without bistability: Worms in nematic electroconvection. `http://xxx.lanl.gov/patt-sol/9802003`.

Rotermund, H.H., Jakubith, S., von Oertzen, A. and Ertl, G. (1991). Solitons in a surface reaction. *Physical Review Letters*, 66:3083-3086.

Sendiña-Nadal, I., Mihaliuk, E., Wang, J., Pérez-Muñuzuri, V., and Showalter, K. (2001). Wave propagation in subexcitable media with periodically modulated excitability. *Physical Review Letters*, 86:1646–1649.

Toyozawa, Y. (1983). Localization and delocalization of an exciton in the phonon field. In Reineker, P., Haken, H., and Wolf, H.C., editors, *Organic Molecular Aggregates: Electronic Excitation and Interaction Processes*, pages 90–106. Springer-Verlag, Berlin and Heidelberg.

Vanag, V. K., and Epstein, I. R. (2001). Pattern formation in a tunable medium: The Belousov-Zhabotinsky reaction in an Aerosol OT microemulsion. *Physical Review Letters*, 87:1–4.

Wang, J., Kada, S., Jung, P. and Showalter, K. (1999). Noise driven avalanche behaviour in subexitable media. *Physical Review Letters*, 82:855–858.

Wuensche, A. (2004). Self-reproduction by glider collisions: The beehive rule. In Pollack, J., Bedau, M. A., Husbands, P., Ikegami, T., Watson, R. A., *Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, Boston, USA, 12–15 September 2004*, pages 286–291. MIT Press, Cambridge, MA.

Wuensche, A., and Adamatzky, A. (2006). On spiral glider-guns in hexagonal cellular automata: Activator-inhibitor paradigm. *International Journal of Modern Physics*, 17:1009–1026.

Part IV

Discussion

15

A Turing Test for Emergence

Fabio Boschetti and Randall Gray

15.1 Introduction

Dealing with *complex* systems presents a particular challenge to many traditional engineering approaches. The pertinent assumption inherent in these approaches is that component parts of a system can be neatly partitioned and that their interactions have limited, predictable effects. This assumption is not always tenable, and it has an impact both on the degree of overall control attainable and on the robustness of the resulting systems. A traffic controller does not need to give exact instructions to each vehicle on the road and a treasurer does not need to control each single business in a country; rather they both provide general guidelines which aim at a desired global outcome; they both rely on the local organization inherent in road traffic and business interactions to account for local details. Similarly, we would like a designer to specify broad guidelines in order for a complex system to act according to a general requirement. Since the inherent organization we wish to exploit is often a dynamical and stable configuration, a system designed to capitalize on this may also display a robustness and adaptivity which is currently beyond our engineering abilities.

In the parlance of complex systems science, the global outcomes arising from broad guidelines on a system's components, including robustness and adaptivity, are often defined as emergent features. As design inevitably requires a trial-and-error process, it is natural to expect that our community will have to develop methods to:

- Detect emergent features when they arise.
- Categorize them in order to understand what classes of processes arise as a result of different initial conditions.
- Experiment with various configurations in order to optimize the emergent processes.

Experimentation is something which is often carried out via computer simulation; however, computers are perfect examples of a 'traditional' engineering apparatus, and consequently display the very same features (lack of robustness and adaptivity and a requirement for detailed instructions) which we are trying to circumvent. In this chapter we explore this apparent paradox and ask what kind of emergent features can

be generated (and thus modelled) in a computational framework. We show that this question directly relates to the other two items listed above, i.e., to the experimental detection of emergence and its classification.

15.2 Background

The concept of emergence evolved to capture our intuition that when a large number of entities interact, the resulting system can display features and behaviours which are not displayed by the individual constituents. The human body possesses behaviours and functions which are not expressed by our individual cells; metals show properties not displayed by individual atoms; societies undergo dynamics which transcend individuals. Basic examples can also be modelled very easily on a computer; famously, Conway's Game of Life (see Gardner 1970) shows how very simple local rules generate features whose dynamics is not explicitly coded in the algorithm. Examples are so ubiquitous in Nature that some scientists suspect that all structures we see 'emerge' from underlying simpler levels (Laughlin and Pines 2000).

Nevertheless, emergence raises considerable intellectual and scientific challenge. Despite a vast literature, going back several decades (Corning 2005), no agreement can be reached on a definition, nor on a framework for its study, nor on whether emergence is a 'real' natural phenomenon or merely a by-product of our perception or a convenient way to make sense of processes otherwise too hard to comprehend (Crutchfield 1994b; Rabinowitz 2005).

Why should a process which appears so obvious and easy to model prove so hard to define and conceptualize? The fundamental reason is that in an emergent process it is very hard to discriminate 'who does what.' When I decide to listen to music, is it my 'emergent' self which takes the decision or my cells? My body depends on cellular activity for its functioning, so cells must be the controlling entities. However, no cell decides to listen to music since listening to music is not something cells 'do.' This leads straight into old and unsolved philosophical problems of causality, determinism, and freewill.

Crucially, this is also a technological problem. Today, probably for the first time in history, technological developments in many applications depend on an understanding of emergent phenomena. Advances in information technology, epidemiology, ecosystem management, health science, just to name a few, depend on approaches which go beyond traditional reductionism and address the understanding of how emergent properties arise, what they 'do,' and how they can be controlled.

It thus seems natural that when we ask whether emergence is 'real' or merely lies 'in the eyes of the observer,' or whether emergence is a distinct process of its own or encompasses different processes among which we are not yet able to discriminate, the answer needs to account for what these processes 'do.' In other words, we have to account for causal relationships and causal power. It may appear that we are trying to address a slippery problem (emergence) via one which is even more slippery (causality). This does not need to be so if we constrain what we mean by causality and adopt an 'operative' definition. Following Pattee (1997) and Pearl (2000), we associate causal

power with control: a process has causal power if, by acting upon it, we can change the effects it produces. Pattee (1997) describes this very simply:

> Useful causation requires control. . . .. Clearly it is valuable to know that malaria is not a disease produced by "bad air" but results from Plasmodium parasites that are transmitted by *Anopheles* mosquitoes. What more do we gain in these examples by saying that malaria is caused by a parasite. . .? I believe the common, everyday meaning of the concept of causation is entirely pragmatic. In other words, we use the word cause for events that might be controllable. In the philosophical literature controllable is the equivalent of the idea of power. In other words, the value of the concept of causation lies in its identification of where our power and control can be effective. For example, while it is true that bacteria and mosquitos follow the laws of physics, we do not usually say that malaria is caused by the laws of physics (the universal cause). That is because we can hope to control bacteria and mosquitos, but not the laws of physics.

Building from this observation and from the work of Crutchfield (1994b, a), Shalizi (2001), and Rabinowitz (2005), among others (Bickhard 2000; Campbell 1974; Darley 1994; Bedau 1997; Andersen et al. 2000; Emmeche et al. 2000; Kauffman 2000; Goldstein 2002; Wiedermann and van Leeuwen 2002; Atay and Josty 2003; Stannett 2003), we propose to discriminate among three types of emergence, depending on increasing level of 'causal' power: pattern formation, intrinsic emergence, and causal emergence.

Despite the philosophical halo of the above discussion, our aim is utterly practical. In a scientific culture in which understanding is increasingly synonymous with computer modelling, we ask what forms of emergence can be studied by simulation and what we can gain from doing so. We will see that computational and 'causal' barriers are strongly related. This may lead to new insights into the limitations and future of the computer modelling of complex processes.

15.3 Formal Logic and Computation

There is an equivalence between the workings of formal grammars, logical systems, and computation (Turing 1936; Penrose 1989; Chaitin 1997; Ord 2002). All these start from some fundamental set of strings (starting symbols, axioms, or input data) and a set of rewriting rules (production rules, rules of inference, computer instructions); they generate outputs (strings, theorems, and computational results), which are obtained by transforming the a priori set via the rewriting rules.

In a formal system, true statements are almost always either theorems or tautologies (Kurt Gödel demonstrated that there are true statements which are not accessible from the axioms and rules of logic. These true statements are not theorems since they are not derived from the axioms). Given a set of axioms and inference rules, these statements necessarily follow and are true for all possible scenarios and cannot be otherwise. Consequently, these statements do not provide any information about the real world [Any information such a string may seem convey is a result of correspondences

we see (or think we see) between the real world and the fundamental system and is wholly dependent on our perception of these correspondences.] An example clarifies the concept: the statement '*it's raining*' may be true today and may or may not be true tomorrow; it depends on its agreement with the vagaries of the real world. Assessing whether the statement is true or not provides information about the real world. Pythagoras' theorem, in contrast, is true independently of Nature's vagaries, it must be true and always will be true. The fact that Pythagoras' theorem is useful to us and matches our perception of reality is due to the clever choices of the basic axioms of geometry. It is because of the appropriateness of axioms collected in Euclid's work that the properties of triangles match our perception of reality.

Given Euclid's axioms and our rules of mathematical reasoning, Pythagoras' theorem is an inevitable consequence. It helps us to understand Nature better by simplifying geometrical considerations, by putting place holders in our geometrical thinking so we do not always need refer back to the axioms, and it helps us to communicate this understanding, but it does not provide any information which is not already implicit in the our axioms and rewriting rules. Theorems are transformations of information, not new information. In some sense, all the theorems of Euclidean geometry could be compressed, with no loss of information, into the basic axioms and inference rules (this is formally proved in Chaitin (1997) and is the base for Kolmogorov-Chaitin's complexity measure). It could reasonably be argued, though, that any decompressor which could reproduce the theorems of Euclidean geometry through its decompression would need to be at least as complex as a mathematician and, like a mathematician, would stand a reasonable chance of passing the Turing test (which we discuss later).

The PCs on our desks are equivalent to a finite tape Turing Machine (TM), an abstract and general computational device commonly employed in theoretical computer science. Because the execution of a TM is equivalent to the application of production rules in a formal grammar and to proof in a formal system (Turing 1936), it follows that the result of running a TM is equivalent to a theorem or a valid string: the results are independent of reality.

It thus also follows that the outputs of any of our computer models are similarly dictated by their initial state and the rewriting rules embodied by the program. [Technically, this is correct provided that the PC does not allow for interaction with the outside world; see Wiedermann and van Leeuwen (2002) and van Leeuwen and Wiedermann (2001a)]. A computer model transforms the information contained in its input via its coded algorithm, but does not generate information. Clearly, a model's output helps our finite mental capability to see consequences of what we coded (which at times we cannot envisage), but its truth status and relevance to the real world is limited to the truth and relevance of the user code and the input fed to the computation. No actual information about the real world is produced by a simulation. Information is generated solely by the writing of the code and the choice of the input. In this way, our choices about how we model a system are much like Euclid's choices and the comparison of the results of our simulations to what we observe in Nature tells us about the appropriateness of the rules we implement and the input we choose.

15.4 Algorithms and Physical Laws

In our perception of reality, causality manifests itself as physical laws. (Conversely, a physical law can represent both causal relations and mere correlations, from which arises the philosophical dilemma behind causality. For the purpose of our discussion it is important to stress that causality can be represented only as a physical law, such as "for every action there is a corresponding equal and opposite reaction.") Our computational representation of physical laws involves algorithms which are essentially transformation rules (sequences of instructions). Since we have seen that transformation rules of this sort are constrained to produce results which are members of a set which is totally determined by these rules and the initial conditions, we have to conclude that the running of algorithms which represent physical laws can only produce similarly deterministic results. Any physical law (rule) which an algorithm can generate must already be implicit in the physical laws (rules) represented in the coded algorithm. No new physical law (or representation of it) can be generated by modelling.

When faced with the questions, "Can genuinely novel causal laws emerge from lower level causal laws?" or "Can causal laws which transcend the causal power of their constituents exist in Nature?," we can envisage two possible answers:

1. Emergent, genuinely novel, causal laws cannot exist and are only apparent and perceived as such because of the limitations in the representation we use.
2. Emergent causal laws must arise via natural processes which are nonalgorithmic, fundamentally different from the workings of a formal logic system and consequently not computable in classical sense.

15.5 Three Levels of Emergence

In this section we examine three levels of emergence often discussed in the literature. Our analysis focuses on the relative causal power of the emergence features they can generate.

15.5.1 Pattern Formation and Detection

Pattern formation captures the most intuitive view of emergence. The interaction of low-level simple entities, leading to symmetry breaking, generates a coordinated behaviour; this is expressed by patterns which are novel and identifiable as such by an external observer. "The patterns do not appear to have specific meaning within the system, but obtain a special meaning to the observer once (and if) he/she is able to detect them. When this happens, the patterns become part of the tool-box the observer can employ to describe and study the process" (Crutchfield 1994b). Examples include the Game of Life discussed above, spiral waves in oscillating chemical reactions, convective cells in fluid flow, and fractal structures in fractured media.

For the purpose of our discussion, pattern formation does not, in itself, imply causal power. Let us consider the Game of Life and the emergent gliders. Detecting their presence is useful for an observer to comprehend the effect of the local rules, to highlight

the potentially universal computational capability of the system and possibly to devise a language able to compress their description (Shalizi 2001; Rabinowitz 2005). The question relevant to our discussion is whether the gliders can 'do' something or are simply 'passive' expressions of internal dynamics; can we exert any causal control on the gliders? What should we do to affect the behaviour of the gliders?

The obvious answer is that we could manipulate the Cellular Automata (CA) local rules. This however acts at the lower level (the CA cells) not at the level of the gliders, which are still merely a representation of our manipulation of the local rules. Can we act on the gliders themselves? We believe that this can happen only via rewriting the CA code, i.e., via an external intervention and a complete redesign of the system. We discuss this more extensively in Section 15.6.3. For now we suggest that pattern formation, per se, does not imply causal power.

15.5.2 Intrinsic Emergence

Intrinsic emergence refers to features which are important within the system because they confer additional functionality on the system itself. These emergent features may support global coordination-computation behaviour like the motion of a flock of birds or stock market pricing (Crutchfield 1994b). Examples with immediate relevance to modelling are minority game models (Arthur 1994, 1998): agents must take local decisions on actions which result in an economic outcome, but they are not able to communicate, so they have no information about other agents' behaviour. If they identify an emergent feature providing information about the global dynamics of the population's economy, then they can use this measure to decide what actions to take (Boschetti 2005). This feature now acts as an avenue for global information processing and provides the system with the possibility of coordinated behaviour. Clearly, the agents' behaviour influences the global measure, but now the global measure affects the behaviour of the agents by determining their future actions. Self-referentiality becomes a fundamental ingredient for complex dynamics and intrinsic emergence.

Discriminating whether intrinsic emergence implies causal control is more challenging and is surely not as clear-cut as for pattern formation. In a real world we could externally affect the stock market (e.g., with some sort of governmental intervention) thereby changing indirectly the dynamics of the agents, who would respond to the sudden external change by altering their future behaviour. This intervention is not possible in the case of pattern formation described above, since we cannot intervene, e.g., on a convective cell without acting directly on the molecules' motion. In the case of a simulation, we could affect the future behaviour of the model by changing the values of the emergent feature (market), without having to reprogram the code. However, this is not fully satisfactory since, in a classic Turing machine, no interaction with the computation is allowed and, consequently, the distinction between algorithm and input data is blurred.

15.5.3 Causal Emergence

The relation between emergence and causality has been studied under the term 'downward causation' or 'strong emergence' (Heylighen 1991; Bickhard 2000; Goldstein

2002). Roughly, 'a feature is emergent if it has some sort of causal power on lower-level entities.' Like all topics involving causality, this is a subject open to considerable controversy (see Rabinowitz 2005). Here we refer to it as 'causal emergence' to highlight the fact that we employ the weaker definition of causality involving control and consequently our conclusions do not necessarily generalize to the global problem of downward causation. Another suitable name could be emergence of control.

With causal emergence we define the arising of structures on which we can exert direct control without manipulating, nor concerning ourselves with, the lower-level constituents. As an example, we assume again that the ultimate cause of human behaviour lies in the biochemical process arising at a molecular and cellular level. Suppose I want to ask my friend Jim to play some music for me. I can do so by addressing him directly, for instance by speaking or by writing a message. Once a message is received, my friend will employ his biological machinery to accept the invitation, but I do not need to concern myself with it. I do not need to reprogram complicated instructions into Jim's cellular substratum. For all practical concerns, my friend acts as an entity with emergent causal power.

15.6 Modelling Causal Emergence

In the previous section we proposed subdividing emergence into three classes depending on the causal power of the features each can generate, ranging from pattern formation, which generates features with no causal power; to intrinsic emergence, displaying limited, indirect causal power; to causal emergence, empowered with full causal power.

In Section 15.4 we claimed that the generation of causal power cannot be modelled, since an algorithm cannot produce novel rules. If this statement is correct, then we deduce that while we can model pattern formation, and we may or may not be able to model intrinsic emergence (depending on whether we allow for interaction with data rather than instructions), we should not be able to model causal emergence. If true, this is quite a bad piece of news, since a large component of research on complex systems today is carried out via computer modelling, and emergence is considered to be a crucial ingredient of complex systems.

This is a potentially important claim. For a claim to be meaningful, however, it has to be relevant and falsifiable. In this section we discuss why this claim is relevant to current scientific investigation by addressing applications to biological and ecological modelling, Artificial intelligence, artificial life, and the mining of large scientific datasets. This leads us along the difficult path of falsification via a variant of the Turing test, applicable to emergence processes. A full discussion of the falsification of this claim requires addressing much subtler issues of the philosophy of science and metamathematics which are beyond the scope of this chapter, but which we touch upon briefly in the final discussion.

15.6.1 Is This Relevant?

Pattern formation is usually considered the most trivial form of emergence. Nevertheless, its relevance to our scientific enquiry is beyond doubt. An inspiring exposition

on the relevance of intrinsic emergence to an understanding of Nature can be found in Crutchfield (1994b), to which we refer the reader. Here we discuss the possible relevance of the concept of causal emergence. As mentioned above we distinguish causal emergence from downward causation in this work. Ample discussion of the related concept of downward causation can be found in Andersen et al. (2000).

In the wake of our discussion in Section 15.4, the relevance of causal emergence depends essentially on whether we believe uncomputability can be found in Nature. On this topic, the scientific community is broadly divided into two groups. The first group, by far the larger, believes that uncomputability exists only in the abstract world of formal logic and pure mathematics, not in the natural world. A common justification of this view is that no example of uncomputability has so far been detected in Nature nor is there a specific need to include it in our descriptive tools. A smaller community believes that uncomputability can be found in Nature. Among these we can cite Penrose's famous claims about the supercomputability of the human brain (Penrose 1994, 1989; Stannett 2003; Kellett 2006). According to Penrose we can easily envisage real implementations of the abstract concept of a Turing machine, so there is no reason to believe that uncomputability cannot be generated in Nature. For a further discussion on this topic see also Calude et al. (1995) and Cooper and Odifreddi (2003). A natural observation for supporters of the latter view is that, if all tools enabling us to study Nature are based on computation (i.e., algorithms), then it follows that no uncomputable process can be detected. This observation leads to a possible third view of the problem, according to which Nature may or may not include uncomputable processes, but we will never be able to detect or access them because of the inherent limitation in the language we use to interpret it. We come back to this possibility later.

15.6.2 Biological/Ecological Modelling

The idea underlying any computer modelling is to create a virtual laboratory where a researcher can perform experiments and scenario testing which would be impossible, impractical, or too costly to carry out in the real world. The relevance of these experiments depends on how well the virtual laboratory resembles the real world. Nineteenth century physics has taught us that perfect accuracy is beyond our reach (e.g., Heisenberg's uncertainty principle), and this teaching is well accepted today. Nineteenth century mathematics has taught more fundamental concerns (e.g., Gödel's theorem), which, curiously, are more easily dismissed.

Considerable experience in engineering, physics, and chemistry has shown immense practical benefits and, when the general limitations are carefully accounted for, has proved how useful computer modelling can be. When porting the approach to biological and ecological modelling, it becomes tempting to employ the same method for studying processes like evolution, adaptation, and creation of novelty and diversity. However, we believe that these processes involve the same causal emergence we discussed above, and it thus becomes necessary to ask whether the virtual laboratory has a similar functional relation to the real world as that enjoyed by physical systems. The same question can be framed as follows: to what extent can a biological agent

be modelled within the same framework that is used to model nonliving objects and processes?

To give a practical example of where the challenge may reside, it is useful to remember that a crucial concept in biological and ecological studies is the existence of multiple levels of organization (cells → organs → individuals → communities → species → ecologies, etc.).

According to our current understanding, these structures self-organize [they do not follow explicit external direction templates (Kauffman 2000)] and are linked by two-way (upward and downward) interactions. Many real world problems (e.g., ecological and renewable resource management) depend on our understanding (i.e., modelling) of this supposedly spontaneous generation of organization and two-way interactions. Obviously, the more complex the questions we ask, the more complicated the models we have to develop and the more levels of organization we may need to include in the model. For example, in a fishery management problem we may want to study how individual fish organize themselves in schools or how individual fishermen organize a fishing fleet. This represents one level of organization. If the specifics (or the scale) of the problem requires it we may also need to model how schools of different fish interact or how a school of fish interacts with a fishing fleet; this represents a second level of organization. In our model we can design a set of rules (a module) which controls the behaviour of the individual fish and vessels and a set of modules for the behaviour of fish schools and the fleet. However, if our purpose is to understand how these multiple levels arise and interact, then we would like the dynamics of the different levels to be shared or at least related. In principle we may want to code a single module (of the lower level) and see how higher levels of organization arise as a result; after all, this is what we conjecture happens in Nature.

Here, in our opinion, a fundamental discontinuity is revealed. In order to model this nesting of organization, the schools and fleets need to be more than mere patterns arising form the lower level; they have to be able to 'do' something. In particular, they have to be able to causally interact with other entities. Following our discussion in Section 15.6, we equate this to asking whether we can exert control on the system without needing to 'refer back' or manipulate the rules controlling the individual fish and vessels. In other words, as we are able to ask our friend to play some music (without needing to concern ourselves about his 'lower level,' local, biochemical rules) by merely interacting with him at a higher level, similarly, we would like to be able to exert control on a school or fleet without having to concern ourselves with the lower-level rules governing them. If we cannot do that, then we must conclude that the school/fleet system is merely a pattern, which we can identify and analyse, but which does not have any causal power. Thus the question is, Can we exert such control?

15.6.3 Artificial Life and Artificial Intelligence—A Turing Test for Emergence

We believe the answer to the previous question is a negative. We also believe that this is merely conjecture and that it cannot be proved. We also believe that this issue is highly debatable since it depends mostly on potentially different interpretations of causal control, as we discuss in this section.

In Section 15.5.1 we expressed our opinion that the gliders in the Game of Life are mere patterns with no causal power, and we asked ourselves whether we can interact with the gliders without recoding their local rules. Answering this is not trivial, mostly because it depends on how much 'purpose' is placed in the original local rules.

We explain what we mean by 'purpose' with an example. Let us take a flocking model (Reynolds 1987). Birds fly in flocks by ensuring that they maintain certain constraints on the position of one with respect to another. Suppose we now place an obstacle on the route of the flock. The flock will circumvent the obstacle. It thus appears that we were able to exert control on the behaviour of the flock; the flock appears to have causal power. However, we ask ourselves whether the flock has actually done anything which was not explicitly coded in the lower-level rules. After all, all the flock did was to maintain flight by following a lead bird which avoided the obstacle. Is there any emergent behaviour in this? Is there any causal power which was not purposely coded.

Can we causally control the flock in any other way? Reasonable arguments can be given in both the affirmative and negative in answering this question. Interestingly, this is not particularly important. Let us consider once again my friend Jim playing some music. It is beyond our current understanding to discriminate to what extent our verbal instruction interacts with his underlying biochemistry. Similarly, it is beyond our current knowledge to grasp how our higher-level invitation (spoken request) is processed at his lower (biochemical) level for the task to be carried out. For our discussion, what matters is only our perception of causal control on Jim's behaviour. By analogy, we are led to conclude that in the Game of Life or flocking example, what matters is the perception by which we believe we can exert causal control over the higher-level emergent features. Does it look as if those features possess causal control? Does it look like they do more than the limited number of behaviours purposely encoded in the local rules? Do system entities behave as if they were autonomously interacting with external processes and respond accordingly?

These new questions have the flavour of a 'Turing test for emergence.' Famously, the Turing test [for related variations, see the series of papers contained in *Minds and Machines* (Saygin et al. 2000; Sterrett 2000)] was designed to circumvent the difficult question of defining what intelligence is and to detect when a computer can be said to have achieved it (one of the original purposes of artificial intelligence at its very conception). Turing suggested testing whether a human (an intelligent agent) was able to discriminate blindly between another human (another intelligent agent) and a computer. Should he/she not be able to do so, then we should conclude than the computer and the human act as intelligently as each other and therefore they are both similarly intelligent. Following analogous reasoning, we conceive an 'emergent' version of the test and we ask whether a process empowered with autonomous causal emergent properties (a human) can discriminate between another causal emergent process and a computer program. Should he/she not be able to do so, then we should conclude that the computer displays causal emergence.

We are not actually suggesting that the test be carried out in earnest. Rather, we would like to refer to and build upon the vast body of work (both conceptual and practical) carried out on the Turing test over several decades and extend some of the conclusions which may be relevant to the study of emergent processes and computer

modelling. In this regard, note that intelligence is itself often considered an emergent feature of the processing occurring in a nervous system. If we accept this view, then the 'Turing test for emergence' can be seen as a generalization of the traditional Turing test. Consequently extending the discussion of the traditional Turing test to emergence becomes more than merely exploiting an imaginary analogy.

The traditional Turing test has been subjected to considerable theoretical discussion and criticism. Nevertheless, practical implementations of the Turing test are carried out annually in the form of the Loebner Prize (Wikipedia 2007). So far, it is widely accepted that improvement in the test performance over the years has not been particularly significant and 'passing the test' does not seem to be a likely short-term outcome. The entire artificial intelligence community has, therefore, revisited its own role, scope, and measure of success. Far from being a proof, this observation does somehow reinforce our conjecture that modelling causal emergence via computer simulation should, at the very least, not be taken for granted.

On a more positive note, this suggests a reason for the Complex System Sciences (CSS) community building more closely on the extensive experience accumulated along the difficult path followed by artificial intelligence. After a few decades of pessimism, a new breeze of optimism can be felt in both the artificial life and artificial intelligence communities. This renewed confidence is not based on the infrastructure of logical programming or the complications of expert systems (as in the past), nor on hopes of supercomputability brought to us by quantum computing. Rather it depends on more down-to-earth, often biologically inspired, approaches. As an example, in a series of papers (van Leeuwen and Wiedermann 2000, 2001b,a, 2003; Wiedermann 2000; Wiedermann and van Leeuwen 2002; Verbaan et al. 2004), van Leeuwen and Wiedermann show formally that agents interacting with their environment have computational capabilities which supersede classic computation. There are a number of reasons why interacting agents can achieve these acrobatics: they run indefinitely (as long as the agent is alive), they continuously receive input from a (potentially infinite) environment and from other agents (unlike a classic machine for which the input is determined and fixed at the beginning of the calculations), they can use the local environment to store and retrieve data, and they can adapt to the environment. In particular, the agents' adaptation to their environment means that the 'algorithm' within the agents can be constantly updated, and in van Leeuwen and Wiedermann (2003) it is shown how supercomputability can arise from the very evolution of the agents. Moreover, in an interactive machine, the traditional distinction among data, memory, and algorithm does not apply, which results in more dynamical and less specifiable computational outcomes (Milner 1993). Other classes of relatively down-to-earth machines which seem to guarantee a breaking of classic computation barriers include fuzzy Turing machines (Wiedermann 2000).

Today human-computer interactions are standard in a large number of applications. Usually, these are seen as enhancing human capabilities by providing the fast computation resources available to electronic machines. Should we see the interaction in the opposite direction, as humans enhance the computational capabilities of electronic machines? van Leeuwen and Wiedermann (2000) speculate that today personal computers, connected via the web to thousands of machines worldwide, receiving inputs

via various sensors and on-line instructions from users, are already beyond classic computers. Today, sensors monitor several aspects of the environment routinely and some have even been installed on animals in the wilderness (Simonite 2005). Can we envisage a network computing system in which agents (computers) interact with the environment via analogue sensors, receive data from living beings, and instructions from humans to deal with unexpected situations? Could this be the way forward to understand emergence?

More intriguingly, could these systems potentially already sit on our desks?

15.6.4 Data Explosion and Scientific Data Mining

In a recent issue of *Nature* (Butler 2006; Muggleton 2006; Szalay and Gray 2006), the picture was drawn of a near future when improved instrumentation and extensive sensing will provide us with exponentially increasing quantities of data for scientific enquiry. This implies more information, but at a considerable cost. It promises more and better information about a vast range of things, from space to ocean depths, from ecologies to the human body, from genomes to social behaviour. However, this explosion of data may go beyond our ability to process and analyse it. Unravelling new mysteries of Nature will then be jeopardized by something as mundane as lack of time and resources. It is hypothesized that this will be circumvented by clever software able to supervise the instrumentation, detection of new interesting patterns, and possibly use of rule extraction algorithms that uncover new processes and biophysical or social laws—a very difficult task, but (supposedly) merely a technological one.

This picture relies on two assumptions:

1. All natural processes we may wish to study or detect are algorithmic.
2. The process which allows us to understand and study Nature is also algorithmic.

Neither of these assumptions has been proved and both are open to debate. The first statement was discussed above. The second one requires some clarification. First, a computational system which scans a dataset in order to find patterns of interest must by definition, be algorithmic. Similarly, a system which, upon detecting a pattern, performs some rule extraction in order to attract our attention and suggest an interpretation also needs to be algorithmic. It seems evident that any algorithm capable of sifting through a stream of data and picking out just those novel patterns which are of interest to a human being, is more than a few steps along the way to passing the Turing test. Similarly, the second of the two systems bears a remarkable resemblance to the halting problem. An algorithmic system cannot, by definition, process a nonrecursive language, from which it follows that if Nature displays a nonalgorithmic process, it will not be detected by a fully automated computational system.

It is interesting to note that the rigors of formal logic apply not only to computational systems, but to the broader scientific method as well. The scientific method requires that experiments be reproducible. This implies that an experiment needs to follow a quite detailed and rigorous procedure in order to be replicated by different observers in inevitably different experimental settings. Basically, an experiment

is reduced to an algorithm (Stannett 2003, p. 122), and consequently scientific experimentation suffers the very same limitation of formal logic and computer systems, and thus is, by itself, unable to detect truly emergent processes. Curiously, the same desire for a rigorous, quasi-algorithmic approach affects scientific communication, with scientific journals often requiring a quasi-algorithmic way of writing. However, it is often suspected that the large leaps in scientific understanding are fired by a brilliance which may be nonalgorithmic. While further considerable work needs to be done to understand this creative process, it seems that overrelying on formal logic not only to model but also to detect and analyse Nature may come with the risky consequence of preventing us from seeing the very processes we want to discover.

15.7 Conclusions

Our aim is by no means to suggest that computer modelling is a purposeless activity. Rather, it is to suggest that clarity is needed to discriminate the means (computer modelling as a tool) from the aim (acquiring knowledge about Nature). In this framework, confusing the means with the aim equates to carrying out a scientific program (including experimental and formal analysis) in the virtual world of a computer model as if it were the 'real world' and then extending the 'virtual' results to the 'real' natural world, under the assumption that the two are, to some degree, isomorphic. Here the old Chinese saying "if all you have is a hammer, everything looks like a nail" nicely highlights possible dangers and could be translated as 'if all you have is a computer, everything looks computational.'

We can thus summarize our proposed guidelines as follows:

1. Care should be taken to discriminate among: the processes which are 'naturally' amenable to computer modelling; the processes which are numerically or theoretically intractable (large combinatorial problems, NP-hard problems, chaotic problems) but for which useful approximations can be found (either in terms of nonoptimal solutions or large-scale approximations); and processes which may be fundamentally intractable.
2. The widely accepted conjecture that intractable problems do not exist in Nature should (at least) be carefully studied, rather than accepted dogmatically.
3. The rigors of the algorithmic approach do not apply only to the world of formal systems and computer languages. Scientific investigation (the iterative testing of hypotheses) is also subject to these constraints due to its algorithmic nature. Recently, a new scientific tendency is to call for a freer and more creative way of reporting and discussing science. Complex system science, which naturally mixes experts ranging from pure mathematics to social science, seems to be in a particularly fortunate development stage for thorough exploration of the potential for reintroduction of artistic and other creative contributions to science.

Acknowledgements

This research was carried out as a part of the CSIRO Emergence Interaction Task, http://www.per.marine.csiro.au/staff/Fabio.Boschetti/CSS_emergence.htm.

References

Andersen, P. B., Emmeche, C., Finnemann, N. O., and Christiansen, P. V. (2000). *Downward Causation*. Aarhus University Press, Aarhus, Denmark.

Arthur, W. (1994). Inductive behaviour and bounded rationality. *The American Economic Review*, 84:406–411.

Arthur, W. (1998). Modeling market mechanism with evolutionary games. *Europhysics News*, 29:51–54.

Atay, F., and Josty, J. (2003). On the emergence of complex systems on the basis of the coordination of complex behaviors of their elements. Santa Fe Institute Working Paper, 04-02-005.

Bedau, M. A. (1997). Weak emergence. In Tomberlin, J., editor, *Philosophical Perspectives: Mind, Causation, and World*, volume 11, pages 375–399. Blackwell Publishers, Oxford.

Bickhard, M. H. (2000). Emergence. In Andersen, P. B., Emmeche, C., Finnemann, N. O., and Christiansen, P. V., editors, *Downward Causation*, pages 322–348. University of Aarhus Press, Aarhus, Denmark.

Boschetti, F. (2005). Improved resource exploitation by collective intelligence. In Zerger, A. and Argent, R. M., editors, *MODSIM 2005 International Congress on Modelling and Simulation.* Modelling and Simulation Society of Australia and New Zealand, December 2005, pages 518–523.ISBN:0-9758400-2-9.

Butler, D. (2006). 2020 computing: Everything, everywhere. *Nature*, 440(7083):402–405.

Calude, C., Campbell, D. I., Svozil, K., and Stefanescu, D. (1995). Strong determinism vs. computability. In Depauli-Schimanovich, W., Koehler, E., and Stadler, F., editors, *Downward Causation*, pages 115–131. Kluwer Academic, Dordrecht.

Campbell, D. T. (1974). Downward causation in hierarchically organized biological systems. In Ayala, F., and Dobzhansky, T., editors, *Studies in the Philosophy of Biology*, pages 179–186, University of California Press, Berkeley.

Chaitin, G. (1997). *The Limits of Mathematics: A Course on Information Theory & Limits of formal reasoning*. Springer, New York.

Cooper, B., and Odifreddi, P. (2003). Incomputability in nature. In Cooper, S. B., and Goncharov, S. S., editors, *Computability and Models*, pages 137–160. Kluwer Academic, Dordrecht.

Corning, P. (2005). The re-emergence of emergence: a venerable concept in search of a theory. In *Holistic Darwinism: Synergy, Cybernetics, and the Bioeconomics of Evolution*. University of Chicago Press, Chicago.

Crutchfield, J. (1994a). Is anything ever new? Considering emergence. In Cowan, G., Pines, D., and Melzner, D., editors, *Complexity: Metaphors, Models, and Reality*, SFI Series in the Sciences of Complexity XIX, pages 479–497. Addison-Wesley, Redwood City.

Crutchfield, J. P. (1994b). The calculi of emergence: Computation, dynamics, and induction. *Physica D*, 75:11–54.

Darley, V. (1994). Emergent phenomena and complexity. In Brooks, R., and Maes, P., editors, *Proceedings of Artificial Life IV*. MIT Press, Cambridge, MA.

Emmeche, C., Koppe, S., and Stjernfelt, F. (2000). Levels, emergence, and three versions of downward causation. In Andersen, P. B., Emmeche, C., Finnemann, N. O., and Christiansen, P. V., editors, *Downward Causation*, pages 13–34. University of Aarhus Press, Aarhus, Denmark.

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "Life." *Scientific American*, 223:120–123.

Goldstein, J. (2002). The singular nature of emergent levels: Suggestions for a theory of emergence. *Nonlinear Dynamics, Psychology, and Life Sciences*, 6(4).

Heylighen, F. (1991). Modelling emergence. *World Futures: The Journal of General Evolution*, 31(Special Issue on Emergence, G. Kampis, editor):89–104.

Kauffman, S. (2000). *Investigations*. Oxford University Press.

Kellett, O. (2006). A multi-faceted attack on the busy beaver problem. Master thesis, Rensselaer Polytechnic Institute, Troy, New York.

Laughlin, R., and Pines, D. (2000). The theory of everything. *Proceedings of the National Academy of Sciences*, 97(1):28–31.

Milner, R. (1993). Elements of interaction. *Communications of the ACM Archive*, 36(1):78–89.

Muggleton, S. (2006). 2020 computing: Exceeding human limits. *Nature*, 440(7083):409–410.

Ord, T. (2002). Hypercomputation: Computing more than the Turing machine. Technical report, University of Melbourne.

Pattee, H. (1997). Causation, control, and the evolution of complexity. In Andersen, P. B., Emmeche, C., Finnemann, N. O., and Christiansen, P. V., editors, *Downward Causation*. University of Aarhus Press, Aarhus, Denmark.

Pearl, J. (2000). *Causality: Models, Reasoning and Inference*. MIT Press, Cambridge, MA.

Penrose, R. (1989). *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Vintage, London, Melbourne.

Penrose, R. (1994). *Shadows of the Mind: A Search for the Missing Science of Consciousness*. Oxford University Press, Oxford.

Rabinowitz, N. (2005). Emergence: An algorithmic formulation. PhD thesis, The University of Western Australia.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Saygin, A., Cicekli, I., and Akman, V. (2000). Turing test: 50 years later. *Minds and Machines*, 10(4):463–518.

Shalizi, C. (2001). *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Michigan, Ann Arbor.

Simonite, T. (2005). Seals net data from cold seas. *Nature*, 438:402–403.

Stannett, M. (2003). Computation and hypercomputation. *Minds and Machines*, 13(1):115–153.

Sterrett, S. (2000). Turing's two tests for intelligence. *Minds and Machines*, 10(4):541–559.

Szalay, A., and Gray, J. (2006). 2020 computing: Science in an exponential world. *Nature*, 440(7083):409–410.

Turing, M. A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265.

van Leeuwen, J., and Wiedermann, J. (2000). The Turing machine paradigm in contemporary computing. Technical Report UU-CS-2000-33, Institute of Information and Computing Sciences, Utrecht University.

van Leeuwen, J., and Wiedermann, J. (2001a). Beyond the Turing limit–evolving interactive systems. In Pacholski, L. and Ruzicka, P., editors, *Theory and Practice of Informatics*, pages 90–109. Springer-Verlag, Berlin.

van Leeuwen, J., and Wiedermann, J. (2001b). A computational model of interaction in embedded systems. Technical Report UU-CS-2001-02, Institute of Information and Computing Sciences, Utrecht University.

van Leeuwen, J., and Wiedermann, J. (2003). The emergent computational potential of evolving artificial living systems. *AI Communications*, 15:205–215.

Verbaan, P., van Leeuwen, J., and Wiedermann, J. (2004). Lineages of automata—a model for evolving interactive systems. In Karhumaki, J., Maurer, H., Paun, G., and Rozenberg, G., editors, *Theory Is Forever*, pages 268–281. Springer-Verlag, Berlin.

Wiedermann, J. (2000). Fuzzy computations are more powerful than crisp ones. Technical Report V-828, Prague University.

Wiedermann, J., and van Leeuwen, J. (2002). The emergent computational potential of evolving artificial living systems. *AI Communications*, 15(4):205–216.

Wikipedia (2007). Loebner prize—Wikipedia, the free encyclopedia, available at: http://en.wikipedia.org/wiki/loebner_prize. [Online; accessed 25-January-2007].

Index

ACO, *see* algorithm, ant colony optimization
adaptability, 4, 42, 52, 247, 249, 255
adaptive enterprise, 6
adaptive resonance theory, 263
advanced materials, 73
agent, 293
  active visualization component, 295
  actuators, 78
  adaptation, 359
  application timeline, 299
  belief, 89
  boid, 306
  communication, 62
  control, 64
  control input, 84, 89
  control policy, 91
  cooperation, 70
  data visualization, 292, 298
  embedded, 52, 68
  impact, *see* impact function, 86
  information-particle, *see* infoticle
  infoticle, 302
  intelligent, 293
  local, 70
  local communication, 299
  local perception, 299
  memory, 299
  mobile, 52, 64
  motion model, 84, 89, 93
  negotiation, 299
  observation model, 90
  path, 84
  perception-action loop, 9, 33
  prioritized acceleration allocation, 306
  robotic, 89
    see robot, 54
  semiautonomous, 51
  sensor/actuator model, 83, 89
  sensors, 52, 78
  situated, 299
  software, 113
  state model, 84
AIN, *see* artificial immune network
AIS, *see* artificial immune system
algorithm
  ACO-DRS, 60
  algorithmic system, 360
  ant clustering, 310
  ant colony optimization, 60, 257
    pheromone, 60, 257
  approximation, 249
  clonal selection, 263
  CLONALG, 255, 263, 264
  clustering, 249
  communication costs optimization, 232
  dead reckoning scheme, 60
  decentralized decision, 86
  decentralized negotiation, 77
  decentralized resource allocation, 225
  distributed dynamic programming, 60
  distributed optimization, 81
  estimation of distribution, 107
  evolutionary, 106, 136
  flocking, 307
  genetic, *see* genetic algorithms
  gradient field, see gradient field, 53

HEFT, 264, 266
heuristic, 247
deterministic operation, 249
immunocomputing, 276
input data, 354
list scheduling, 249
processor selection, 249
task duplication, 249
task insertion, 249
task prioritization, 249, 259
negative selection, 254, *see* artificial immune system
nonrecursive language, 360
parallel, 206, 264, 317
pattern recognition, 275
permutation mask approach, 254
place-and-route, 197
Q-learning, 106
random guided search, 249
random walk, 249
randomized search, 248, 249
roulette wheel selection, 230
rule extraction, 360
shortest path, 199
simulated annealing, 247
cooling schedule, 250
neighbor selection, 250
singular value decomposition, 276
SOTL, 43
sotl-platoon, 42
tabu search, 247
transformation rules, 353
ANN, *see* artificial neural networks
anomaly detection, 247, 254
ant-based foraging, 310
architecture
service-oriented, 217
ART, *see* adaptive resonance theory
artificial immune network, 261
artificial immune system, 247, 271
abstraction, 257
antibody, 258
clonal generation, 261
clonal selection, 247, 250
hypermutation rate, 260
receptor editing, 261
selection probability, 264
danger theory, 247
elimination, 247
foreign agents, 247
H-cells, 262
host, 247
immune network, 247
immunological crossover, 258
innate immune system, 250
learning, 250
lymphoid organ, 258
memory, 250
modeling, 257
negative selection, 247, 250, 287
pattern recognition, 250
positive selection, 287
recognition, 247
S-cells, 262
self-nonself discrimination, 287
artificial intelligence, 355
applied, 294
cybernetics, 22
artificial life, 344, 355
artificial neural networks, 179, 247, 285
artificial neuron, 257
error back propagation, 285
assortative noise, 11
autonomic computing, 6
autonomic informatics, 6
autonomy, 21, 42, 51, 116, 178, 201, 220, 252, 293, 299, 358
axiom, 351

Bayes rule, 94
Bayesian filtering, 90
behavior, 350
abnormal, 263
asymptotic, 12
autonomous
see autonomy, 179
centering, 307
chaotic, 12
collective, 154, 293
collision avoidance, 84, 306
coordinated, 5, 354
discontinuous, 142
dynamic, 298
emergent, 51, 292, *see* emergence
evolved, 137
flocking, 292
global, 3, 42, 354
local, 283

microfluidic, 153
nondeterministic, 4, 318
periodic, 12
reactive, 113
rule-based, 293
self-organized, 19
self-regulatory, 5
soliton-like, 325
spatiotemporal, 342
stabilization of, 307
stable, 8
statistical, 7
swarming, 154, 293, 305
symmetrybreaking, 10
velocity matching, 306
Belousov-Zhabotinsky medium, 26, 326
binary phase shift key (BPSK), 70
bio-inspired engineering, 19, 179
biology, 179
antibody, 275
antigen, 275
apoptosis, 271, 272
autoimmunization, 271
blood vessel, 147
central nervous system, 110
chemokine, 152
cytokine, 271
differential reproduction, 250
DNA-based self-localizations, 325
ecological system, 329
autotroph, 329
heterotroph, 329
generation, 250
genotype, 250
immune system, 271
immunological response, 157
molecular recognition, 275
muscle, 108
mutation, 255
natural selection, 250
neuro-immune-endocrine modulation, 272
ontogenesis, 118
phenotype, 250
phenotypic trait, 118
phylogenesis, 118
population, 250
prey-predator population, 344
protein, 157
selection pressures, 5
tubulin microtubules, 325
vertebra, 108
bounded rationality, 220

CA, *see* cellular automata
causality, 350
circular, 8
cell (agent)
damage, 70
orientation, 70
cell matrix, 179
bootstrap, 194
C lines, 182
cell, 180
reconfigurable, 180
clock, 182
configuration, 185
continuous, 210
control, 203
D lines, 181
mode, 182
neighbors, 180, 181
simulator, 188
target cell, 191
tools, 203
truth table, 181
wires, 191
Cell Matrix Corporation, 180
cellular automata, 12, 154, 282, 293, 310, 328, 354
deterministic, 285
glider, 13, 331, 353
collisions, 13, 331
homogeneous dynamics, 329
nondeterministic, 284
qualitative taxonomy, 12
central pattern generator, 110
CFG, *see* grammar, context-free
Chapman-Kolmogorov equation, 90
chemical sensing, 150
chemical wave, 286, 326
wave-fragment, 326
traveling, 329
chemotaxis, 154
communication, 354
acosutic, 151
asynchronous, 79
bandwidth, 77
bounded delay, 80

channel, 81
cost, 220
mobile code, 222
remote communication, 222
delay, 78, 80
diffusion-mediated, 151
electromagnetic, 151
frequency, 78
intercluster, 102
interprocessor, 265
mobile code, 218
network, 82
noise/signal ratio, 127, 134
overhead, 13, 249
physical medium, 82
protocols, 52
rate, 82
dynamic, 86
remote, 218
scalable, 6, 78
stigmergy, 151
structure, 81
topology, 9
ultrasonic, 63, 66
wireless, 71
complex systems, 22, 223, 349, 355
internal structure, 292
complexity, 22, 77
ϵ-machine, 22, 27
computational, 9
Kolmogorov-Chaitin, 352
morphological, 26
statistical, *see* statistical complexity
composition operator, 83, 96
cumulative, 84
computation, 5, 351, 352
algorithmic approach, 13, 361
biologicallyinspired, 247
biomolecular immunocomputer, 271
combinatorial, 361
DNA-based, 171
nature-inspired, 247, 328
environmental flux, 247
nonclassical, 328
NP-hard, 361
reaction-diffusion model, *see* reaction-diffusion
self-organizing, 6, 12
soft computing, 248
supercomputability, 356
uncomputability, 356
computational intelligence, 272
computational mechanics, 11
computer security, 253
computing
grid computing, 219
grid computing model, 225
on-demand, 217
service-oriented, *see* service-oriented paradigm
controller
central, 51, 52
coordination, 5
CPG, *see* central pattern generator

DAG, *see* scheduling, directed acyclic graph
damage, 52
critical, 57
diagnosis, 57, 72
evaluation, 70
extent, 52
location, 58
noncritical, 57
severity, 57, 72
system response, 51
data
communication, 69
data acquisition, 55
data mining, 253
agent-based, 295
ant-based, 310
data visualization, 291
self-organizing, 292
dataset
temporal, 299
dead-reckoning, 70
decentralized data fusion, 90
decision
collective, 138
decentralized, 86
global, 78
initial, 79
local, 79, 158, 354
optimal, 78
perturbation, 85
refinement of, 79, 86
local, 79

modular, 84
vector, 79, 85
decision making, 77
decentralized, 77
distributed, 77
defects
manufacturing, 203
runtime, 204
diagnostics, 58, 161
diffusion, 158
diffusive capture, 150
digital logic, 177, 178
digital signal processing, 68
dissipative structures, 9
distance metrics, 261
Euclidean distance, 261
Hamming distance, 261
Manhattan distance, 261
distributed processing, 52
distributed system, 217
engineering, 6
downward causation, 22, 354
drag force, 152
DRS
see algorithm, dead-reckoning scheme, 60
dynamic systems initiative, 6
dynamical hierarchy, 22
dynamical system, 20
attractor, 23
chaotic, 12
spatial structure, 25
bifurcation, 24
parameter, 23
chaotic regime, 7
control parameter, 7
degrees of freedom, 24
deterministic, 26
fast foliation, 24
fast short-lasting component, 7
initial conditions, 7
low-dimensional, 7
macrostate, 130
manifold
slow, 24
stable, 23
unstable, 23
microstate, 130
mode
stable, 7
unstable, 7
ordered regime, 7
phase space, 12, 21, 110
quasi-periodic orbits, 13
slow long-lasting component, 7
submanifold, 24
symmetry, 25
symmetry breaking, 138

EC, *see* evolutionary computation
ecological atlas, 282
EDA, *see* algorithm, estimation of distribution
El Farol Bar problem, 223
embryonics, 180
emergence, 5, 19, 29, 139, 179, 325, 349
causal, 351, 355
intrinsic, 5, 351, 354
perceptual, 297
strong, 354
visual, 298
emergent behavior, *see* behavior, emergent
emergent intelligence, 116
enslaving principle, 7, 24
entropy, 9, 21, 47, 90, 95, 140
as objective function, 91, 144
Boltzmann entropy, 9, 127, 129
conditional, 27
disorder, 131
index of, 131
entropy rate, 9, 131
generalized, 10
excess entropy, 10, 28, 131
Gaussian, 95
joint, 27
minimization, 91
normalized index, 132, 141
posterior, 91, 95, 96
production, 9, 21
reduction, 21
equilibrium, 21
error-correcting encoding, 5
evolution, 5, 10, 106, 116, 134, 250, 283, 336, 356
evolutionary algorithms, 106
evolutionary computation, 247
evolutionary design, 8
intrinsic selection criteria, 9
task-specific objective, 9
evolvable hardware, 179

exciton, 325
extensible markup language, 112

fabrication process driver, 205
fault detection, 247, 254
FIN, *see* formal immune network
Fisher information, 94
fluid velocity, 152
formal immune network, 271
  affinity, 273
  antigen, 275
  antobody, 276
  apoptosis, 273
  autoimmunization, 273
  cell, 273
  cytokine FIN, 272
  epitope, 274
  innate immunity, 273
  inner invariant, 274
  pattern, 275
  self-organization, 273
  training, 277
formal system, 13, 351
FPGA, 178
fractal structure, 353
friction
  isotropic, 108
fuzzy systems, 247

Gödel's theorem, 13, 356
GA, *see* genetic algorithms
Game of Life, 310, 344, 350, 353
game theory, 222
genetic algorithms, 8, 106, 136, 179, 248, 287
  control parameter, 250
  crossover, 257
  crossover frequency, 250
  fitness function, 250
  fitness gradient, 123
  fitness landscape, 106
  gene, 106
  linear chromosome, 106
  mutation, 250, 257
  mutation frequency, 250
  population size, 250
  recombination, 250
  termination condition, 250
genetic programming, 10, 106
  algorithmic implementation, 114
  grammar-based, 107
  learning mutation strategy, 107
genotype, 26
Ginzburg-Landau equations, 24
Ginzburg-Landau theory, 7
GP, *see* genetic programming, 109
gradient field, 60
  class, 61
  dynamic, 62
  implementation, 60
  modification, 61
  stability, 62
  training, 60
  visualization, 72
grammar
  context-free, 110
  context-sensitive, 107
  formal, 351
  grammatical evolution, 107
  learning probabilistic, 107
graph coloring, 255
grid, 217
grid toolkits, 221

halting problem, 360
hardware compilation, 201
hardware swapping, 202
hardware time sharing, 202
Heisenberg uncertainty principle, 356
Hessian, 81, 86
heterogeneity, 220, 266
hierarchical clustering, 102
homeostatic resilience, 4
hybrid systems, 13
hydroacoustics, 288
hydrophysical field, 282

immune network, 256, 263
  formal, *see* formal immune network
  stimulatory, 257
  suppressive, 257
immune system, 247, 250
  adaptive, 250
  affinity, 252, 255
  antibody, 251
    idiotope, 256
    paratope, 256
  antibody-antigen binding, 251
  antigen, 251, 255

epitope, 256
apoptosis, 255
autoimmune diseases, 254
B-cells, 251
cellular immunity, 251
clonal expansion, 255
clonal selection, 255
danger theory, 254
differentiation, 252
gene, 263
gene library, 263
homeostatic regulation, 255
homoral immunity, 251
lymphocyte, 252
memory, 252
necrosis, 255
negative selection, 253
pathogen-associated molecular pattern, 251
pattern-recognition receptors, 251
proliferation, 252
self-nonself discrimination, 253
T-cells, 251
immunochip emulator, 279
immunocomputing, 271
impact
damage, 52
detection, 54
location, 57
severity, 57
impact detection, 55
multiple, 60
impact function, 83, 96
impact space, 83, 96
size, 84
task-specific, 84
independent component analysis, 33
inductive reasoning, 220, 223
inference rule, 351
information, 352
assurance, 288
bottleneck, 28
dynamics, 11
flocking, 309
flow, 21, 33
gathering, 78
active, 89
processing, 5, 271
transfer, 9
information theory, 26
information-driven evolutionary design, 9, 123
infrastructure, 51
integration, 27
interactive machine, 359
intrinsic information, 27
intrusion detection, 254, 272
attack, 279
recognition time, 282

Jacobian, 23, 94
just-in-time compilation, 202

Kalman filter, 94
Kohonen map, *see* self-organizing map

Landauer's principles, 21
Law of requisite variety, 8
Least-square method, 283
light bullet, 325
light-sensitive molecular array, 325
liquid crystal, 326
local interactions, 5, 51, 128, 179, 292, 353
agent-environment, 78
between robots, 135
hydrodynamic, 153
interagent, 293
intercellular, 272
minimization of conflicts, 11
mutualistic, 344
pairwise agent, 300
recursive, 292
strength of, 7
strong, 7
traffic, 41
weak, 7
localization, 325
breather, 325
excitation state, 331
refractory state, 331
logical system, 351
logistic map, 29

machine learning, 247
magic polygons, 208
management
self-management, 218
MEMS technology, 169
microenvironment, 148
biophysical properties, 154

chemical sources, 149
spatial structure, 149
microtubule, 325
migration decision problem, 218, 222
minimum entropy production principle, 9
minority game, 354
morphogenesis, 19
multiagent system, 51, 77, 293
adaptive, 52
algorithms, 52
central controller, 51
collaboration, 77, 97
complex, 52
control, 92
cooperation, 293
coordinated motion, 138
coordination, 42
coupling, 78, 80, 85
of oscillators, 110
decentralized, 293
decentralized control, 6, 92
decision problem, 78, 97
distributed, 52, 54
dynamical, 89
impact detection, 54
joint belief, 91
modularization, 83
multi-robot system, 6, 78, 127
alignment, 139
path planning, 84
resource management, 220, 221
size, 84
multiinformation, 27
mutual information, 27
mutualistic excitation, 328

nanoscale, 147
nanotechnology, 73, 206
negative database, 254
negotiation, 220
network communication, 55
network security, 253
network traffic, 217
nomadic service, 219
nonassortativeness, 11

O-self-organization, 30
object localization, 92
objective function, 77, 91, 248
based on order parameter, 8
coordination of distributed actuators, 9
curvature, 80, 86
efficiency of communication topology, 9
efficiency of locomotion, 9
entropy, 91, 95
fitness function, 262
generalized, 83, 96
maximization of information transfer, 9
minimization of heterogeneity, 9
partially separable, 83, 85
shared, 77
stability of multiagent hierarchies, 9
observer, 30
coarse-grained, 31
external, 5, 353
fine-grained, 31
observer-based measure, 31
perfect, 30
ODE, *see* open dynamics engine
open dynamics engine, 112
operations research, 248
optical scattering, 153
optimal gradient, 5
optimization, 77, 349
asynchronous, 81
communication costs, 222, 227
complexity of, 77
condition, 79
control parameter, 8
convergence, 81, 82
distributed, 81
of traffic flow, 41
particle-swarm optimization, 306
suboptimal approximation, 248
Oregonator equations, 326
organization information, 30
oscillating chemical reactions, 353
oscillon, 325

partial separability, 83
particle animation, 293
pattern formation, 3, 5, 19, 24, 51, 351, 353
collective motion, 308
comet pattern, 304
excitation, 329
chaotic, 331
quasi-chaotic, 329
exploitation, 5

exploration, 5
global pattern, 3
long-term zoning, 308
macroscopic, 7
quark pattern, 304
short-term clustering, 308
side-winding, 10
spatial, 30
star pattern, 304
swarming, 309
variability, 13
pattern recognition, 247, 253, 275, 277
phase transition, 7, 22, 128, 132
predictive information, 11
predictor
active, 227
efficiency, 228
function, 226
selection, 228
set, 229
type, 227, 229
probabilistic learning model, 106
probability density, 90
conditional, 89, 93
Gaussian, 93
probability distribution
posterior, 94
prior, 94
probability measure, 26
uniform distribution, 266
processing
local, 52

Rayleigh-Ritz theorem, 279
reaction-Diffusion, 25
reaction-diffusion, 325
redundancy, 197
reinforcement learning, 222
resource
availability, 248
capability, 248
heterogeneity, 248
resource allocation
agents, 221
distributed, 220
electronic market models, 221
resource facilitator, 220
resource scheduling, 221
robot
actuator
distributed, 10
autonomous, *see* autonomy
biomedical applications, 148
chassis, 134
communication, 62, 66
control, 105
cooperative, 53
gripper, 134
in SHM systems, 64
inchworm, 64
locomotion, 64, 70
side-winding, 10
microscopic, 147
action, 149
Brownian motion, 148
communication, 149
control of, 149
distributed control, 148
locomotion, 149, 153
power generation, 153
sensor, 149
thermal noise, 148
microsurgery, 169
motion, 134
navigation, 60–62, 69
neural network control, 127, 136
optical rangefinder, 64
panoramic camera, 92
self-organized movement, 54
sensor, 92
bearing, 92
failure, 163
traction, 135
snakelike, 105
video camera, 70
robotics
collective, 127
modular, 10
swarm, 6
robustness, 4, 11, 42, 51, 52, 144, 247, 249, 252, 349

SC-self-organization, 29
scalability, 4, 51, 78, 180, 255
schedule length, 262
scheduling, 248, 249
constraints, 248
resource, 248

temporal, 248
decisions, 248
directed acyclic graph, 264
communication to computation ratio, 265
dynamic, 248
flow shop, 248, 264
hybrid, 264
job, 248
completion time, 248
partitioned, 248
precedence constraints, 249, 259
sequence, 263
job shop, 248, 263
lower bound solution, 263
macro-dataflow graph, 264
multiprocessor, 248
near optimal, 249
NP-complete, 249
NP-hard, 247, 263
objectives, 248
makespan, 248, 264
normalized schedule length, 265
resource utilization, 248
response time, 248
overheads, 248
performance metrics, 258
preemption, 263
quality, 249
schedule length, 248
static, 248
task allocation, 259
task graph, 249, 264
time complexity, 249
sea surface temperature, 283
self-assembly, 5
self-modifying, 177
self-organization, 51, 252
applications, 3
autocatalytic process, 5
avalanche effect, 138
conformist principle, 133
constraints, 8
convergence, 133, 279
coordination, 3, 128
design, 4
design space, 7
energy exchange, 3
equilibrium, 4, 138, 307
thermodynamic, 22
far from equilibrium, 7
fixed point, 24
information dynamics, 12
information exchange, 3, 128
information transfer, 4
information-theoretic approach, 21, 30, 51
interactions, 343
measure of, 21, 129
multiagent, *see* multiagent system
negative feedback, 9, 128
noise barrier, 134
O-self-organization, 30
order parameter, 7, 22
phenomenon of, 19
positive feedback, 9, 128
principles of, 128
random fluctuations, 128
resistance to noise, 143
SC-self-organization, 29
self-reinforcing process, 134
spatial, 26
stability, 139
symmetry, 22
symmetry breaking, 10, 129, 353
theory of, 3
self-organized criticality, 5
self-organizing map, 19, 30, 58
data vectors, 59
evaluation, 59
training, 58
self-organizing traffic lights, 42
self-referentiality, 354
self-repair, 52, 179
self-replication
efficiency, 201
exploded grid, 199
main grid, 199
parallel, 201
self-testing, 193
sensor
design, 52
sensors
communication, 56
damage, 56
elastic wave, 56
embedded, 51
piezoelectric, 54
polymer (PVDF), 66
PZT, 67, 68

service discovery, 225
service level agreement, 219
service-oriented paradigm, 217
SHM, *see* structural health monitoring
simulation, 108, 349, 352
  environment, 231
  multi-robot system, 134
  parameters, 231
  physically-based, 155
  rigid body dynamics, 112
singular value decomposition, 275
Snakebot, 105
  actuators, 105
    correlation, 110
  genetic representation, 109
  locomotion gait, 105
    adaptation, 119, 120
    evolution, 116
    generality, 120
    rectilinear, 116
    sidewinding, 116
  morphology, 108
SOA, *see* architecture, service-oriented
soliton, 286, 325
SOM, *see* self-organizing map
spanning tree, 60
  minimum, 60
spiral wave, 331
statistical Complexity, 11, 28
Stirling's approximation, 132
stochastic analysis, 155
stochastic process, 27
strange loop, 8
structural health monitoring, 51
submicron, 204
supercell, 198
  differentiation, 198
  genome, 198
  interconnection, 199
  isolation, 198
superconductivity, 24
synchronization, 41, 79
  of oscillators, 8
Synergetics, 24
synergetics, 7

tangled hierarchy, 8
Taylor expansion, 94
testing
  circuitry, 197
  parallel, 197
  runtime, 197
theorem, 351
three-dimensional fabrication, 205
TM, *see* Turing machine
traffic
  average trip waiting time, 45
  density, 41, 45
  flow, 41
  green wave, 41
  management system, 41
    green light district, 44
    traffic light, 42
  modelling, 41
  simulator, 42
    moreVTS, 44
triangulation, 69
truth table, 181
  deserialization, 208
  serialization, 208
Turing machine, 13, 352
Turing test, 352
  for emergence, 358

verification, 13
virtual hardware, 202
virtual organization
  on-demand, 217
visualization, 167, 291
  agent-based, 295
  behavioral animation, 301
  cellular ant method, 309
  dynamic animation, 301
  feature space, 294
  image space, 294
  information flocking method, 305
  infoticle method, 300
  multiagent, 295
  multidimensional scaling, 309
  particle system, 301
  scientific, 291
  temporal grouping, 301
visualizer, 54, 71

wafer-scale integration, 205

XML, *see* extensible markup language