Guillaume Bourque
Nadia El-Mabrouk (Eds.)

Comparative Genomics

LNBI 4205

RECOMB 2006 International Workshop, RCG 2006
Montreal, Canada, September 2006
Proceedings

Springer

Lecture Notes in Bioinformatics 4205

Edited by S. Istrail, P. Pevzner, and M. Waterman

Subseries of Lecture Notes in Computer Science

Guillaume Bourque Nadia El-Mabrouk (Eds.)

Comparative Genomics

RECOMB 2006 International Workshop, RCG 2006
Montreal, Canada, September 24-26, 2006
Proceedings

Springer

Volume Editors

Guillaume Bourque
Genome Institute of Singapore
Computational and Mathematical Biology
60 Biopolis Street, Singapore 138672
E-mail: bourque@gis.a-star.edu.sg

Nadia El-Mabrouk
Université de Montréal
Département d'Informatique et de recherche opérationnelle (DIRO)
Montréal, H3C 3J7, Canada
E-mail: mabrouk@iro.umontreal.ca

Library of Congress Control Number: 2006932402

CR Subject Classification (1998): F.2, G.3, E.1, H.2.8, J.3

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN 0302-9743
ISBN-10 3-540-44529-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-44529-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11864127 06/3142 5 4 3 2 1 0

Preface

The complexity of genome evolution poses many exciting challenges to developers of mathematical models and algorithms, who have recourse to a spectrum of algorithmic, statistical and mathematical techniques, ranging from exact, heuristic, fixed-parameter and approximation algorithms for problems based on parsimony to probabilistic models, requiring Monte Carlo Markov Chain algorithms for Bayesian analysis.

The annual RECOMB Satellite Workshop on Comparative Genomics (RECOMB Comparative Genomics) is a forum on all aspects and components of this field, ranging from new quantitative discoveries about genome structure and process to theorems on the complexity of computational problems inspired by genome comparison. The informal Steering Committee for this meeting consists of David Sankoff, Jens Lagergren and Aoife McLysaght.

This volume contains the papers presented at the 4th RECOMB Comparative Genomics meeting, which was held in Montreal, Canada, on September 24- 26, 2006. The first three meetings of this series were held in Minneapolis, USA (2003), Bertinoro, Italy (2004) and Dublin, Ireland (2005).

This year, 34 papers were submitted, of which the Program Committee selected 17 for presentation at the meeting and inclusion in these proceedings. Each submission was refereed by at least three members of the Program Committee.

RECOMB Comparative Genomics 2006 had several invited speakers, including: Lars Feuk (The Hospital for Sick Children, Toronto), Tao Jiang (University of California, Riverside), Fiona Brinkman (Simon Fraser University, Burnaby) Liqing Zhang (VirginiaTech, Blacksburg) Thomas J. Hudson (McGill University, Montreal).

In addition to the invited talks and the contributed talks, an important ingredient of the program was the lively poster session.

In closing, we would like to thank all the people who submitted papers and posters and those who attended RECOMB Comparative Genomics 2006 with enthusiasm.

September 2006 Guillaume Bourque and Nadia El-Mabrouk

Organization

RECOMB Comparative Genomics 2006 was organized by the DIRO (Department of Computer Science), the CRM (Centre de Recherches Mathématiques) and the Robert Cedergren Centre at the University of Montreal.

Steering Committee

Aoife McLysaght (Trinity College, Dublin)
Jens Lagergren (Stockholm Bioinformatics Centre & KTH)
David Sankoff (University of Ottawa)

Program Committee Chairs

Guillaume Bourque (Genome Institute of Singapore)
Nadia El-Mabrouk (DIRO, University of Montreal)

Program Committee

Lars Arvestad (School of Computer Science and Communication, Stockholm), Robert Beiko (The University of Queensland, Australia), Anne Bergeron (Université du Québec à Montréal, Canada), Mathieu Blanchette (McGill University), Michael Brudno (Department of Computer Science, University of Toronto), Cedric Chauve (Université du Québec à Montréal, Canada), Avril Coghlan (Trinity College, Dublin), Miklós Csürös (Computer Science Department, University of Montreal), Dannie Durand (Departments of Biological Sciences and Computer Science, Carnegie Mellon University), Niklas Eriksen (Mathematical Science, Chalmers University of Technology and Göteborg University, Sweden), Rose Hoberman (Computer Science Department, Carnegie Mellon University), Tao Jiang (Department of Computer Science and Engineering, University of California, Riverside), Aoife McLysaght (Trinity College, Dublin), Bernard Moret (Swiss Federal Institute of Technology, Lausanne, Switzerland), Laxmi Parida (IBM, Watson Research Center, Yorktown, USA), Yves van de Peer (Department of Plant Systems Biology, Ghent University, Belgium), Ben Raphael (Department of Computer Science & Engineering, University of California, USA), Cathal Seoighe (National Bioinformatics Network, University of Cape Town, South Africa), Eric Tannier (Laboratoire de Biométrie et Biologie Evolutive, Université Claude Bernard, France), Glenn Tesler (Department of Mathematics, University of California, USA), Stacia Wyman (Department of Computer Science, Williams College, Williamstown).

Organizing Committee

Nadia El-Mabrouk (DIRO, University of Montreal)
Guillaume Bourque(Genome Institute of Singapore)
Mathieu Blanchette (McGill University)
Franz Lang (Biochemistry Department, University of Montreal)
Vincent Ferretti (McGill University)
Sylvie Hamel (DIRO, University of Montreal)
Miklós Csürös (DIRO, University de Montreal)
François Major (DIRO, University of Montreal)
Mathieu Lajoie (DIRO, University of Montreal)

External Reviewers

Max Alekseyev
Guy Baele
Malay K. Basu
Mark Chaisson
Matteo Comin
Daniel Dalevi
Melissa Davis
Martin Frith
Neil Jones
Jacob Joseph
Steven Maere
Istvan Miklos
Narayanan Raghupathy
Magnus A. Rosenblad
David Sankoff
Benjamin Vernot

Sponsoring Institutions

Robert Cedergren Centre, University of Montreal
CRM (Centre de Recherches Mathématiques), University of Montreal
"Faculté des arts et des sciences", University of Montreal

Table of Contents

Reconstructing Domain Compositions of Ancestral Multi-domain Proteins

Behshad Behzadi and Martin Vingron

Computational Molecular Biology Department, Max Planck Institute for Molecular Genetics, Ihnestrasse 73, 14195 Berlin, Germany
{behshad.behzadi, martin.vingron}@molgen.mpg.de

Abstract. A model for the evolution of multidomain proteins should encompass not only sequence divergence but also domain duplications and recombination of domains. Given a set of contemporary multidomain proteins from different species together with a species tree, in this paper, we suggest a model for constructing the domain compositions of ancestral multi-domain proteins considering the above evolutionary events.

Keywords: Evolution of Multidomain proteins, Tree reconciliation.

1 Introduction

Many proteins are composed of domains in the sense that certain parts of the protein, the domains, are similar among otherwise different proteins. Domains are usually thought of being contiguous on the primary sequence. Sometimes one thinks of them as forming a compact three-dimensional fold, while in other context a particular primary sequence pattern is the defining feature. For the purpose of this paper it suffices to agree that a domain forms a contiguous stretch on the primary sequence and that it can be found in many different proteins. Since a domain often also fulfills a particular biological function, domains represent a basic structural, functional and evolutionary unit in the world of proteins [13]. Although the size of domains varies widely one may assume an average size on the order of 100 residues [8]. A majority of eukaryotic proteins are made from more than one domain and are called multidomain proteins. Some multi-domain proteins have even more than one hundred domains. It has been observed that the proteins formed from combinations of few domain (less than 100) play a significant role in determination of functions and phenotypes in organisms [15]. Different resources for protein domain identification and the analysis of domain architectures are available (see for example Pfam [4], CATH database [18], or SMART [12]).

We see the determination of the evolution of multidomain proteins as a challenging problem in its own right. To study the evolution of multidomain proteins it is insufficient to consider only sequence divergence (micro-evolutionary events). Rather we need to include as operations domain duplications (macro-evolutionary events) as well as the recombination of domains. Although existing work on the evolution of multidomain proteins makes it clear that all these events contribute to the evolution of these proteins, no approach or model has

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 1–10, 2006.

been proposed which considers all the above operations. In this work, we report on our work in progress about reconstructing the evolution of multidomain proteins with a model considering the three types of events. As we will see the concept of reconciled trees plays an important role in our approach.

The organization of the paper is as follows: In Section 2 we summarize some existing work on the evolution of multidomain proteins. In Section 3 we formalize the problem of reconstructing ancestral multidomain proteins. We also present a general scheme for solving this problem considering sequence divergence, duplications and domain recombinations. Section 4 suggests a scoring measure which we have chosen for the purpose of defining parsimonious scenarios. The last section contains our conclusions and some directions for future work.

2 Past Work

A large amount of research work has been done on the evolution of multidomain proteins. We categorize these efforts into two different lines of research: work on molecular mechanisms and rates of evolutionary events of multidomain proteins, on the one hand, and work on domain-level comparison and evolutionary models of multi-domain proteins, on the other hand. The current effort we would categorize under the second category.

Apic et al. [1,2] have studied the combinations of domains in the three kingdoms of life (Bacteria, Archaea, and Eukaryota). In particular, they showed that there are a large number of combinations that are shared among the three kingdoms. They showed also that there are many combinations common among families of the species, which are specific for a given kingdom. They conclude that in order to create new functions, nature more frequently makes combinations of already existing building blocks (here protein domains) than inventing new ones.

Ye and Godzik [25] have compared the *domain graphs* (nodes are domains and there is an edge between two domains if and only if they co-occur in some protein architectures) of individual organisms with each other. The goal there is to identify the similarities and differences between their domain organizations. They have also identified common and specific domains and combinations of domain in each of the three kingdoms. The point to keep in mind is that different sets of protein architectures may have the same domain graphs.

Kummerfeld et al. [10,11] have studied the role of gene fusion and fission in the evolution of multi-domain proteins. They found that fusion occurs about four times more often than fission. Pasek et al.[17] in separate but similar work recently investigated the question of finding the molecular mechanisms which causes new domain combinations. They have confirmed the major role of gene fusion and gene fission in evolution of multidomain proteins.

Insertions and deletions of complete domains occur frequently and play an important role in the evolution of multidomain proteins. The work of Weiner et al. [20] on deletion of domains in the multi-domain proteins shows that deletions occur domain-wise; in the other terms, in most of the cases some complete (and not partial) domains are lost. In addition, they have shown that in many cases a

deleted domain is a part of larger, deleted fragment of a protein. Aroul-Selvam et al. [3] have observed more complicated types of insertions in which a domain (or multiple domains) are inserted *into* another domain. These types of insertions are not frequent and we do not consider them in this study.

Vogel et al. [24] have studied the relationship between the domain duplication and recombination. They claimed that these events together are the major actors in inventing new domain architectures. Similar studies in [22] have shown that proteins are formed by duplication, sequence divergence and recombination of domains. There is an evidence that certain two-domain and three-domain combinations recur in different protein contexts with different partner domains more than expected. These evolutionary units which are larger than single domains are called *supra-domains* (see [23]).

The order of the domains in multidomain proteins have also been considered to be important in the formation of the proteins. Different mechanisms for the circular permutation of the domains in a protein have been proposed, too [21]. A given pair of domains, co-occurs in a domain architecture nearly always in the same order. So although the order of the domains can be sometimes important for the functionality of the protein, in this paper we do not consider the order of the domains in a protein architecture.

In the second category of the works, we mention two works which are more related to this paper. Björklund et al. [5] have defined a novel protein similarity measure, *domain distance* which is calculated as the number of domains that differ between two domain architectures of two given proteins. They have constructed an evolutionary tree using this domain distance, which has more domain insertions and deletions than the internal repetition and exchange of a domain. While considering a domain-level distance for building evolutionary trees for proteins is a necessary step, it is not sufficient for building reliable evolutionary scenarios. Sequence divergence of the proteins and the different kinds of recombinations should be included in the model as well.

Przytycka et al. [19] have considered two set operations, merge and domain deletions, as the evolutionary events of the multidomain proteins. In this abstraction, domain merge refers to biological mechanisms which combine two or more domains into a new protein. Domain deletion is any process which causes a loss of one of more domains of the protein. In their model the order of domains in the domain architectures is neglected. Although interesting, this approach also ignores sequence divergence and tandem duplication of domains. In the current paper we follow [19] in using a set theoretic abstraction of domain-level events such as merge and deletion; moreover we include the domain duplication and the sequence divergence to the model.

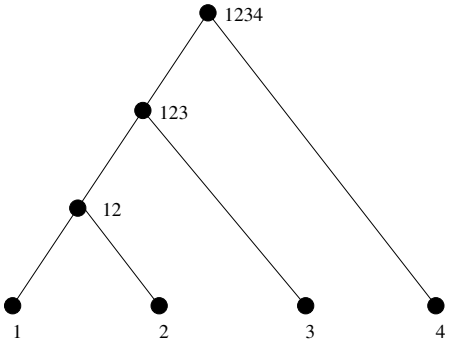
3 Model

In this section we formalize the problem of reconstructing possible ancestral sets of multi-domain proteins that may have evolved into a number of given, contemporary multi-domain proteins. We present a general approach for solving

this problem. For the sake of simplicity, and hopefully without loss of generality, some concepts will be explained only using an example. One example will be developed throughout the exposition.

Let $\mathcal{D}$ denote the set of all domains existing in the contemporary proteins of our study. As stated before, we do not consider the order of the domains in a protein; each protein architecture is represented by a *multiset* (a set where elements may have repetitions) of elements of $\mathcal{D}$. These multisets are denoted by *domain compositions* as they reflect the domain contents of proteins without considering their ordering. The multiset of all domain compositions of proteins of a given species s is called the *domain composition family* of s. The domain composition family of s is represented by a multiset of mulitisets of elements in $\mathcal{D}$. Note that for each contemporary protein, in addition to the architecture (which gives us the domain composition), we have the amino-acid sequence of the protein and its domains. The species tree which relates the species of the study is denoted by $\mathcal{S}$. The set of the leaves of the species tree which denote the contemporary species is denoted by $L(\mathcal{S})$. The problem we address in this paper can be stated as follows: Given a species tree $\mathcal{S}$, and all proteins (amino acid sequence + domain composition) of each species in $L(\mathcal{S})$, find the domain composition family of each of the ancestral species. Figures 1 and 4 show an input (without sequence) and output example of our problem.

Example 1. Our set of species in this example consists of four species, $L(\mathcal{S}) = \{1, 2, 3, 4\}$. The species tree is given in Fig. 1. Let $\mathcal{D} = \{A, B, C\}$ be the set of all protein domains of species 1 to 4. The domain composition families of the species denoted by R_1 to R_4 respectively are shown in Fig. 1.



$$R_1 = \{\{A, B\}, \{A, B, C\}\}$$

$$R_2 = \{\{A, B\}, \{A, B, C\}, \{A, A, C\}\}$$

$$R_3 = \{\{A, A, B\}, \{A, B\}, \{A, C\}\}$$

$$R_4 = \{\{A, C, C\}, \{A, C\}, \{A, C\}\}$$

Fig. 1. The species trees and the domain composition families of the current species (leaves of the tree)

Note that the internal nodes have been labeled 12, 123, and 1234 (according to the leaves of the subtree rooted at that node). In order to be able to identify the individual domains of our example, we relabel the domains of each species in a given order (here left to right). For a domain $X \in \{A, B, C\}$, X^i_j refers to the jth occurrence of domain X, in species i. The above sets are relabeled as follows:

$R_1 = \{\{A_1^1, B_1^1\}, \{A_2^1, B_2^1, C_1^1\}\}$
$R_2 = \{\{A_1^2, B_1^2\}, \{A_2^2, B_2^2, C_1^2\}, \{A_3^2, A_4^2, C_2^2\}\}$
$R_3 = \{\{A_1^3, A_2^3, B_1^3\}, \{A_3^3, B_2^3\}, \{A_4^3, C_1^3\}\}$
$R_4 = \{\{A_1^4, C_1^4, C_2^4\}, \{A_2^4, C_3^4\}, \{A_3^4, C_4^4\}\}$

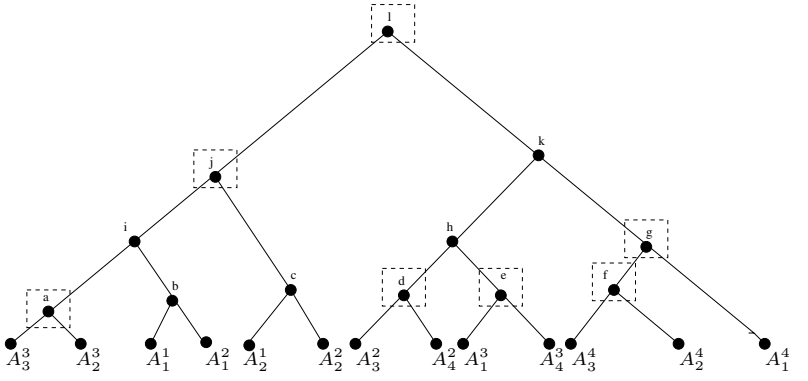
Let us recall that gene trees, are trees which represent the evolutionary relationships of a set of homologous genes. Gene trees are constructed based on the micro-evolutionary events (sequence evolution) operations. Different methods [6,7,14] have been proposed for constructing gene trees (or more generally phylogenetic trees). Very similar to the definition of gene trees, we define *domain trees*, which are the phylogenetic trees inferred from the sequences of protein domains. Obviously, the methods applied for constructing gene trees can be used in the case of domains as well.

Similar to gene trees, the domain trees and species trees may be different in topology because they present evolutionary relations of different entities. This is mainly due to the macro-evolutionary events on domains like domain duplications. We use *reconciled trees* in the context of domain trees in order to identify the minimum number of domain duplications which explain the difference of a domain tree and a species tree. Using a tree reconciliation algorithm [16] and the *lca* (least common ancestor) function one can localize these minimal number of duplications on the species tree. This provides us with the number of copies of each domain present in each of the ancestral species. Hypothetical domain trees for the protein domains of Example 1, are given in Fig 2. The tables correspond to the *lca* function of the nodes of domain trees. Recall that if the lca function of a node and at least one of its children is the same node in species tree, this refers to a duplication at that node.

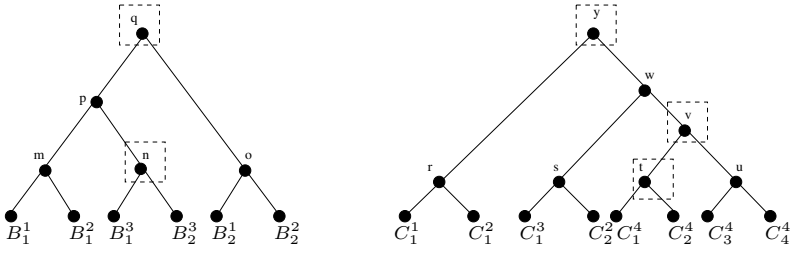
The total number of domains in the domain compositions of the ancestral nodes is given in Fig. 3. The domain trees together with their reconciliation with the species tree provide valuable information about the evolution of the multidomain proteins. Let us make this clearer in our example. We can for example deduce that one of the domains C_1^4 and C_2^4 (in $\{A_1^4, C_1^4, C_2^4\}$) is (most probably) the result of a tandem duplication of the other one. However, For A_1^3 and A_2^3 (in $\{A_1^3, A_2^3, B_1^3\}$) the situation is different. They should be a result of union of two parts of two ancestral proteins. The trees also show us that $\{A_1^1, B_1^1\}$ and $\{A_1^2, B_1^2\}$ should come from an ancestral protein which contains both A and B in its domain composition.

It should be noted that although we have used the same domain name A for the three copies of A in node 12, these three copies can be distinguished by their different children (one of them is the parent of A_1^1 and A_1^2, one other is parent of A_2^1 and A_2^2 and the last one is the parent of A_3^2 and A_4^2). The same fact holds for all of the ancestral domains, because the child parental relations of the domains are provided by the reconciled trees. As a consequence, the set of all domains of an ancestral species can be considered as a set without repetition; this will be used in the next section.

Once the domain copy numbers of the ancestral nodes of the species trees are known, the next step is to reconstruct the domain compositions of the proteins of these species. In this phase, we define a measure of similarity between a



A_1^1	A_2^1	A_1^2	A_2^2	A_3^2	A_4^2	A_1^3	A_2^3	A_3^3	A_4^3	A_1^4	A_2^4	A_3^4	a	b	c	d	e	f	g	h	i	j	k	l
1	1	2	2	2	2	3	3	3	3	4	4	4	3	12	12	2	3	4	4	123	123	123	1234	1234



B_1^1	B_2^1	B_1^2	B_2^2	B_1^3	B_2^3	m	n	o	p	q	C_1^1	C_1^2	C_2^2	C_1^3	C_1^4	C_2^4	C_3^4	C_3^4	r	s	t	u	v	w	y
1	1	2	2	3	3	12	3	12	123	123	1	2	2	3	4	4	4	4	12	123	4	4	4	1234	1234

Fig. 2. Domain trees and the lca function (given in tables) used for reconciling them with the species tree given in Fig. 1. *lca(x)* for a node x in a domain tree is the deepest node x' int the *species tree* such that the set of leaves of subtree rooted at x is a subset of leaves of the subtree rooted at x'. The rectangles refer to nodes that have the same lca value as at least one of its children. These refer to the domain duplications.

parent domain composition family and the two children domain composition families, which estimates the minimum amount of evolutionary events in terms of set operation (domain recombinations) needed to transform the parent domain composition family into each of the child domain composition families.

Two types of set operations are considered in this study: union of domain compositions and deletion of domains. Note that each protein of a child can be uniquely expressed as a union of some non-empty subsets of the parent sets. The domain composition family reconstruction phase is explained in the next section.

But before going to the next section let us summarize our approach into the following three phases:

1) Making domain trees: The sequence divergence (sequence evolutionary events) are considered in this part.

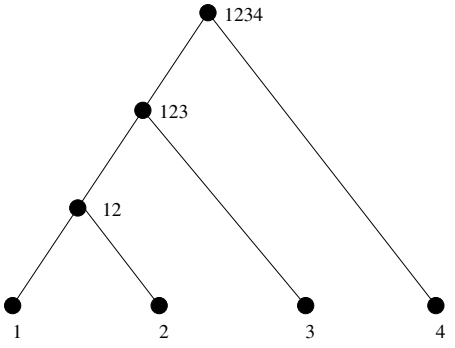


Fig. 3. The total number of domains in domain compositions of ancestral species; these numbers are calculated using the reconciled trees and the domain duplication concept

2) Reconciling domain trees: Domain Duplications are considered in this part. Results are used to determine the number of domain copies in the ancestral nodes. **3) Reconstructing domain composition families** domain-level recombination events such as union and deletions are considered in this part.

4 Domain Composition Family Reconstruction

In this section, we formalize our scoring scheme which is used in reconstructing domain compositions of proteins of ancestral species. Our approach is a bottom-up reconstruction of the species domain composition families. We present a measure of similarity between the domain composition family of a parent node and the ones of its children. Such a measure can be used to define a procedure for reconstructing the domain composition family of a parent node when the domain composition families of the children are given.

We present a set theory formulation of the problem. Consider we have three species A, B and C where A is the parent of B and C. In fact $B = \{B_1, B_2, ..., B_m\}$ and $C = \{C_1, C_2, ..., C_n\}$ where each B_i and C_j is a subset of $\{1, ..., N\}$ for any $1 \leq i \leq m$ and $1 \leq j \leq n$. The domain composition reconstruction family problem can be stated as follows:

Parent Domain Composition Family Reconstruction Problem

Input: Two families (B and C) of subsets of $\{1, ..., N\}$ such that $(\cup_1^m B_i) \cup (\cup_1^n C_j) = \{1, ..., N\}$

Output: A family $\{A_1, ..., A_k\}$ for some $k > 0$ such that
a) $A_i \neq \phi$ for any $1 \leq i \leq k$;
b) $A_i \cap A_j = \phi$ for any $1 \leq i < j \leq k$;
c) $\cup_1^k A_i = \{1, ..., N\}$;
d) $\delta(\{A_1, ..., A_k\}, \{B_1, ..., B_m\}) + \delta(\{A_1, ..A_k\}, \{C_1, ..., C_n\})$ is minimized.

Before presenting the function δ, we need to explain how domain duplicates are interpreted in this formalization. Note that the we are looking for a partitioning of the elements $\{1, 2, ..., N\}$ in A (i.e. no duplicates in A); the reason for this (as explained in the last section) is that we relabel the copies of the same domain in the ancestral node by new different labels. The domains in the children (B and C) will also be relabeled to have the same label as their parent domain in the parent species. For example, for the internal node 12 in our example, $N = 3 + 2 + 2 = 7$. After relabeling, multiple copies of a domain with the same label inside a single child subset (B_i and C_j) are replaced by only one copy of the domain. In fact these duplications have been considered in the tree reconciliation part and we should not reconsider them again.

The function δ which can be defined in different ways should reflect the cost of transforming the protein domain composition family of a father to its children in the species tree. Here the domain level events are considered as set operations: union of compositions and deletion of domains. In our definition this cost is additive on the subsets of child's family; the cost is equal to the sum of the costs of evolutionary events needed to generate each of the child domain compositions. Formally, if $A = \{A_1, ..., A_k\}$ then $\delta(A, \{B_1, ..., B_m\}) = \delta(A, \{B_1\}) + ... + \delta(A, \{B_m\})$. Condition (d) of the output can be rewritten as minimization of $\Sigma_1^m \delta(A, \{B_i\}) + \Sigma_1^n \delta(A, \{C_j\})$. For each single domain composition X ($X = B_i$ for some $i \leq m$ or $X = C_j$ for some $j \leq n$), we define $\delta(A, \{X\})$ as follows:

$$\delta(A, \{X\}) = \underbrace{(|\{i | A_i \cap X \neq \phi\}| - 1)}_{\#\text{ needed unions}} .c_U + \underbrace{(\Sigma_{i:A_i \cap X \neq \phi} |A_i - X|)}_{\#\text{ elements to be deleted}} .c_D \quad (1)$$

The first term refers to the number of compositions in the parent family that are being used in constructing X. The second term is the amount of domain deletions which should be applied on the union of these compositions in order to obtain X. The positive numbers c_U and c_D denote the cost of a union operation and a domain deletion respectively.

So far, we have not found an efficient algorithm to compute this, such that we resort to simulated annealing (SA) [9] for finding the optimal solution of the parent domain composition family reconstruction. We start by a random partition of $\{1, ..., N\}$ and we improve the solution by swapping elements from one subset into another or making new subsets. Although designing a better algorithm for finding the optimal solution for large scale datasets is necessary, we use the SA approach in order to test and confirm our model.

In Figure 4 the reconstructed domain composition families for the ancestral nodes of the species tree of our example is given. Here both union and deletion costs are considered to be 1.

The tree shows that some of the compositions such as $\{A, B, C\}$ and $\{A, C\}$ have existed in the common ancestor of all of the species; these can be regarded as old domain compositions. On the other hand, some other compositions like $\{A, A, C\}$ are more recent.

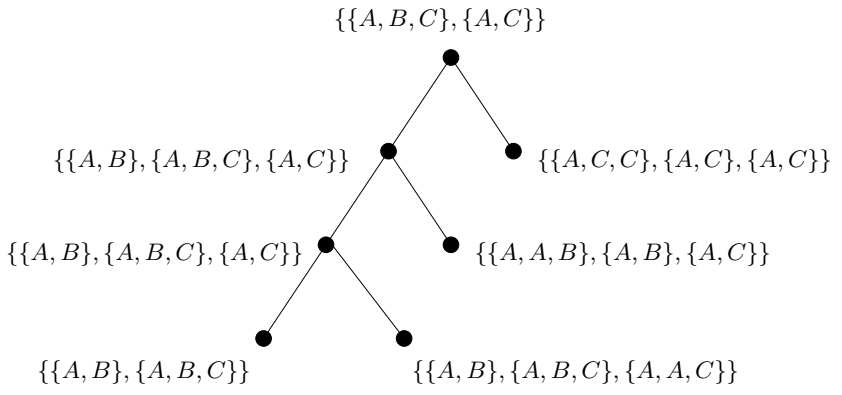


Fig. 4. Domain composition family reconstruction of ancestral species

One should note that our general approach in this paper is not dependent on the choice of score. One can define another similarity score for the domain composition families and then apply the general approach we suggested in the last section.

5 Conclusions and Future Works

In this paper we proposed a general approach for reconstructing the ancestral multidomain protein compositions as well as their evolution. Sequence divergence, domain duplications, and the domain-level recombinations (union of compositions and domain deletions) are included in our model. We want to stress the importance of using the domain trees and their reconciliation in the study of evolution of multi-domain proteins.

This is a work in progress and we are currently working in improving algorithmic aspects of the family reconstruction as well as applying the model on real biological data. Another important future direction of our work is to consider the reliability of the edges of the domain trees, e.g. in the form of bootstrap values, in the scenario computation.

References

1. G. Apic, J. Gough and S. A. Teichmann: An insight into domain combinations. *Suppl. Bioinformatics* 17, S83-S89, 2001.
2. G. Apic, W. Huber and S. A. Teichmann: Multi-domain protein families and domain pairs: comparison with known structures and a random model of domain recombination. *Journal of Structural and Functional Genomics 4, 67-78, 2003.*
3. R. Aroul-Selvam, Tim Hubbard and Rajkumar Sasidharan: Domain Insertions in Protein Structures. *J. Mol. Biol.* 338, 633-641, 2004.

4. A. Bateman, L. Coin, R. Durbin, R.D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E.L. Sonnhammer, D.J. Studholme, C. Yeats, S.R. Eddy: The Pfam protein families database. *Nucleic Acids Res.*, 32, 138-141, 2004.
5. Asa K. Björklund, Diana Ekman, Sara Light, Johannes Frey-Skött and Arne Elofsson: Domain Rearrangements in Protein Evolution. *J. Mol. Biol.* 353, 911-923, 2005.
6. J. Felsenstein: Phylogenies from molecular sequences: Inference and reliability, *Annu. Rev. Genet.* 22, 521-565, 1988.
7. W. Fitch and E. Margoliash: Construction of phylogenetic trees. *Science* 155, 279-284, 1967.
8. Suhail A. Islam, Jingchu Luo and Michael J.E. Sternberg: Identification and analysis of domains in proteins. *Protein Engineering* 8(6), 513-526, 1995.
9. Kirkpatrick S., C. D. Galatt and M. P. Vecchi: Optimization by Simulated Annealing. *Science*, 220, 4598, 671-680, 1983.
10. Sarah K. Kummerfeld, Christine Vogel, Martin Madera, Mary Pacold and Sarah A. Teichmann: Evolution of Multi-domain Proteins by Gene Fusion and Fission. *Proceedings of ISMB'2004.*
11. Sarah K. Kummerfeld and Sarah A. Teichmann: Relative rate of gene fusion and fission in multi-domain proteins. *Trends. in Genetics.* 21(1), 25-30, 2005.
12. I. Letunic, R. R. Copley, B. Pils, S. Pinkert, J. Schultz and P. Bork: SMART 5: domains in the context of genomes and networks. *Nucleic Acids Res.* 34, 257-260, 2006.
13. A. G. Murzin, S. Brenner, T. Hubbard and C. Chotia: SCOP: the structural classification of proteins database. *J. Mol. Biol.* 247, 536-540, 1995.
14. M. Nei: Molecular Evolution Genetics, Columbia University Press, New York, 1987.
15. C.A. Orengo and J.M. Thornton: Protein Families and their Evolution–A Structural Perspective. *Anni. Rev. Biochem.*, 74, 867-900, 2005.
16. R. D. M. Page: Maps between trees and cladistic analysis of historical associations among genes, organisms and areas. *Syst Zool.*, 43, 58-77, 1994.
17. S. Pasek, J.-L. Risler and P. Brézellec: Gene Fusion/Fission is a major contributor to evolution of multi-domain bacterial proteins. *to appear in Bioinformatics*, 2006.
18. F.M. Pearl, C.F. Bennett, J.E. Bray, A.P. Harrison, N. Martin, A. Shepherd, I. Sillitoe, K. Thornton and C.A. Orengo: The CATH database: an extended protein family resource for structural and functional genomics. Nucleic Acids Research 31(1), 452-455, 2003
19. T. Przytycka, G. Davis, N. Song and D. Durand: Graph theoretical insights into evolution of multidomain proteins. *J. Computational Biology* 13(2):351-63, 2006.
20. January Weiner 3rd, Francois Beaussart and Erich Bornberg-Bauer: Domain deletions and substitutions in the modular protein evolution. *the FEBS journal*, 273, 2037-2047, 2006.
21. January Weiner 3rd and Erich Bornberg-Bauer: Evolution of Circular Permutations in Multidomain Proteins. *Molecular Biology and Evolution* 23(4), 734-743, 2006.
22. Christine Vogel, M. Bashton, N.D. Kerrison, C. Chotia, S. A. Teichmann: Structure, function and evolution of multidomain proteins. *Current Opinion in Structural Biology*, 14, 208-216, 2004.
23. Christine Vogel, Carlo Berzuini, Matthew Bashton, Julian Gough, and Sarah A. Teichmann: Supra-domains: Evolutionary Units larger than Single Protein Domains. *J. Mol. Biol.* 336, 809-823, 2004.
24. Christine Vogel, Sarah A. Teichmann and Jose Pereira-Leal: The Relationship Between Domain Duplication and Recombination. *J. Mol. Biol.* 346, 355-365, 2005.
25. Yuzhen Ye and Adam Godzik: Comparative Analysis of Protein Domain Organization. *Genome Research* 14, 343-353, 2004.

Domain Architecture in Homolog Identification

N. Song, R.D. Sedgewick, and D. Durand

Department of Biological Sciences, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. Homology identification is the first step for many genomic studies. Current methods, based on sequence comparison, can result in a substantial number of mis-assignments due to the alignment of homologous domains in otherwise unrelated sequences. Here we propose methods to detect homologs through explicit comparison of domain architecture. We developed several schemes for scoring the similarity of a pair of protein sequences by exploiting an analogy between comparing proteins using their domain content and comparing documents based on their word content. We evaluate the proposed methods using a benchmark of fifteen sequence families of known evolutionary history. The results of these studies demonstrate the effectiveness of comparing domain architectures using these similarity measures. We also demonstrate the importance of both weighting critical domains and of compensating for proteins with large numbers of domains.

1 Introduction

Homology identification arises in a broad spectrum of genomic analyses, including annotation of new whole genome sequences, construction of comparative maps, analysis of whole genome duplications and comparative approaches to identifying regulatory motifs. Currently, sequence comparison methods are widely used to identify homologous genes. These methods assume that sequences with significant similarity share common ancestry, *i.e.* are homologs. However, the existence of multi-domain proteins and complex evolutionary mechanisms pose difficulties for traditional, sequence based methods. A domain inserted into two unrelated protein sequences causes those sequences to have a region of similarity, resulting in mis-assignment of homologs.

To address this issue, researchers frequently also require that the alignment between a pair of potential homologs extend over a large fraction of their lengths [1]. However, the accuracy of this *alignment coverage* heuristic is unknown and there are many examples where alignment coverage makes an incorrect determination of protein homology. For example, human FOXB1 and FOXP3 are known homologs with a significant BLAST [2] e-value of $10^{-8.95}$, but the alignment covers only one fourth of the shorter protein sequence. A heuristic to determine homology based on alignment coverage would fail to recognize these proteins as homologs. Conversely, human KIF5C and mouse BICD2, which have a region of sequence similarity due to a shared HOOK domain, have a BLAST e-value of $10^{-7.26}$ and an alignment coverage of more than half of the length of the

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 11–23, 2006.

shorter sequence. They are not in the same family, but an alignment coverage heuristic would decide that they are. An accurate and automatic method for the identification of homologs remains an open problem.

In this work, we investigate explicit comparison of domain architecture in predicting homology. Complex, multidomain families, such as membrane-bound receptors and cellular matrix adhesion proteins, are characterized by varied domain architectures within a single family. For example, the Kinase domain partners with over 70 different domains in the eukaryotic Kinases. A given member of the Kinase family may contain from one to a dozen domains. Typically, a pair of Kinases has one or more shared domains, but each member of the pair also has domains that do not appear in the other. At the same time, these protein sequences also share domains with sequences not in the Kinase family. The challenge is to determine which aspects of domain architecture (e.g., total number of domains in the sequence, the set of distinct domains, copy number, domain promiscuity) are most informative for separating homologs from unrelated sequences that share a domain.

In order to develop measures of domain architecture similarity we exploit an analogy between domain architecture composition and a problem in information retrieval, namely, determining the similarity of two documents drawn from a corpus. In this metaphor, the word content of a document is analogous to the domain content of a protein sequence and the set of protein sequences under study is analogous to the set of documents in the corpus.

In this work, we adapt information retrieval techniques to domain architecture comparison as a method for identifying multi-domain homologs. We evaluate the effectiveness of several methods on fifteen different protein families by applying each method to test sets composed of positive examples (pairs of proteins both in the family) and negative examples (pairs of proteins with only one member in the protein family). We use this empirical approach to determine:

- whether domain content comparison is, in fact, an effective method for identifying homologs,
- what information about domain content is most informative for this purpose,
- what measure of domain architecture similarity is most effective in identifying homologs.

1.1 Model

A domain is a sequence fragment that will fold independent of context into a protein subunit with specific shape and function. Domains are natural evolutionary units. New domain architectures arise via complex mechanisms such as non-homologous recombination, transposable elements and retrotransposition [3, 4, 5, 6, 7, 8, 9, 10]. About two thirds of proteins in prokaryotes and 80% of proteins in eukaryotes are multi-domain proteins [11].

Domain databases [12, 13, 14, 15, 16, 17] store probabilistic, sequence-based models of protein domains. These models, typically encoded as a Position Specific Scoring Matrix (PSSM) or a Hidden Markov Model (HMM), can be used

to determine the location and type of domains in amino acids sequences. These models can therefore be applied to new sequences about which domain information is unknown.

Using domain databases, a protein can be represented as an unordered list of the domains it contains. For this paper, we compare proteins based solely on this unordered list of domains, ignoring any differences in the linker sequences between domains. Any two instances of the same domain will have some variance at the sequence level. To simplify the comparison, we treat every instance of a domain as identical.

The use of domain architecture composition to define homology has not been previously investigated. A few papers in the literature have considered measures of domain content similarity in other contexts. CDART (Conserved Domain Architecture Retrieval Tool) [18], a web-based resource, presents a list of sequences with similar domain architecture to a given query sequence. Similarity is calculated by counting the number of domains common to the two proteins. A distance metric for domain architecture comparison has been considered in the framework of evolutionary events [19], where distance is defined to be the number of domain insertions and deletions needed to transform one protein to the other. The effectiveness of these methods in determining protein homology was not evaluated.

Now consider again the analogy with document similarity in information retrieval. A simple measure of the similarity of two documents is the number of shared words. Documents that share a significant number of words are more likely to be on the same subject than documents that share few words. Similarly, two proteins that share many domains are more likely to be homologs than sequences with few domains in common. This is the basis of the method used in CDART. Similarity scores based on common domains can take the number of instances of each domain (the copy number) into account (as done by CDART) or can ignore copy number and simply count the number of distinct types of shared domains.

Two long documents are also more likely to share a large number of words than two shorter documents. To correct for this effect, a length correction is used, where length is typically measured by word count. Correcting for the domain count is also useful in the context of comparing protein domain composition. Jaccard similarity, which is commonly used in information retrieval, is appropriate here. It is given by

$$J(p_1, p_2) = \frac{n_{12}}{n_1 + n_2 - n_{12}}, \tag{1}$$

where n_{12} is the number of domains common to both sequences p_1 and p_2, n_1 is the number of domains in p_1, and n_2 is the number of domains in p_2. Using a Jaccard similarity instead of an uncorrected score allows comparisons among a set of sequences with large differences in the number of domains.

These similarity measures treat all words (or domains) equally. However, words in a document are not equally important to determining its subject. The word "big" conveys less information about the subject of document than the word "parsimony," for example. Words should be weighted based on how informative

they are. One measure of the power of a given word to distinguish between subjects is the *inverse document frequency* [20], a measure based on the observation that a word that occurs in very few documents is more likely to differentiate between subjects then a word that occurs in a large fraction of the documents in the corpus. Similarly, some domains are more informative than others. "Promiscuous" domains are domains that occur in many unrelated proteins [11, 21]. Intuitively, promiscuous domains and non-promiscuous domains reveal different amounts of evolutionary information and should be treated differently. We consider several methods for weighting promiscuity.

Since promiscuous domains occur in many unrelated sequences, they are less useful for determining homology than relatively rare domains. We can adapt the inverse document frequency to obtain a weighting system to reduce the emphasis of promiscuous domains on the score. This gives an idf weight for a domain d of

$$w_{\text{idf}}(d) = \log_2 \frac{|\mathcal{P}|}{|\{p|d \in D(p), p \in \mathcal{P}\}|}, \tag{2}$$

where $\mathcal{P}$ is the set of proteins, and $D(p)$ is the multi-set of domains in a protein p. The denominator is the number of proteins in the dataset that contain the domain d.

The frequency of a word in a given document is also an indication of its importance to that document. The word "parsimony" is more likely to be relevant to a document in which it appears five times than to a document in which it occurs only once. This aspect can be expressed by the *term frequency*, the number of times a word appears in a document amortized by the total number of words in that document. In multidomain sequences, term frequency is analogous to domain copy number. We define the term frequency,

$$w_{\text{tf}}(d, p) = \frac{N(d, p)}{|D(p)|}, \tag{3}$$

where $N(d, p)$ is the number of occurrences of the domain d in the protein p and $|D(p)|$ is the number of domains in the protein p.

Typically, words are weighted using the product of w_{tf} and w_{idf},

$$w_{\text{tf-idf}}(d, p) = w_{\text{tf}}(d, p) \times w_{\text{idf}}(d). \tag{4}$$

By comparing the results from idf with tf-idf weighting, we obtain a measure of the importance of copy number to multidomain homology detection. Note that since w_{tf} includes the total number of domains in protein sequence p in the denominator, the tf-idf weight includes a correction for the number of domains in a protein, while the idf weight does not.

We designed a third approach for weighting domains considering how multidomain protein sequences evolve. Both gene duplication and domain shuffling are involved in the evolution of multidomain protein families. Gene duplication results in a pair of related protein sequences sharing a domain, while domain shuffling results in unrelated protein sequences sharing a domain. A domain

that appears in many unrelated sequences is not a strong indicator of homology. However, idf weighting simply measures the number of sequences in which a domain appears without considering whether or not those sequences are related. A domain that appears in many sequences, but always in the same context, could be very informative yet have a low idf weight. For example, if a domain appears in all sequences in a large gene family that arose through repeated gene duplication and does not appear in sequences from any other family, it should be a strong indicator of homology.

A measure of domain promiscuity that captures this idea is the number of distinct domain partners associated with it, where two domains are *partners* if they co-occur in at least one protein. For this reason, we consider weighting a domain by the inverse of the number of distinct domain types with which the domain occurs in sequences. We refer to this weighting method as "distinct partner" weighting:

$$w_{\mathrm{dp}}(d) = \frac{1}{|\{d_i | d_i \in D(p), d \in D(p), p \in \mathcal{P}\}|} \tag{5}$$

Note that the denominator is the number of distinct domain partners of the domain d. This weighting was first developed for studying protein functions in [22], but, to our knowledge, the analogous weighting method has not been used in information retrieval.

Regardless of which weighting scheme is used (idf alone, tf-idf, or distinct partner weighting), we calculated the weighted domain comparison score as follows:

$$S(p_1, p_2) = \sum_i w(d_i, p_1) w(d_i, p_2), \tag{6}$$

where $w(d_i, p_1)$ and $w(d_i, p_2)$ are the weights for domain d_i in proteins p_1 and p_2. The sum runs over all domains, but $w(d_i, p) = 0$ if d_i does not occur in the protein p.

Although the tf-idf and distinct partner weight can give a measure of the importance of different domains, they do not include any correction for the number of domains in a protein and they cannot be used with the Jaccard similarity, which does not apply in the weighted case. This is addressed in information retrieval using the cosine similarity, which is similar to the Jaccard similarity but can be used on either weighted or unweighted domains. For two proteins p_1 and p_2, the cosine similarity is given by

$$C(p_1, p_2) = \frac{\sum_i w(d_i, p_1) w(d_i, p_2)}{\sqrt{\sum_i w(d_i, p_1)^2 \sum_i w(d_i, p_2)^2}}. \tag{7}$$

Treating the protein sequences as vectors in a vector space with number of dimensions equal to the number of domain types, cosine similarity is the cosine of the angle between the two weight-vectors. In the case of unweighted methods the cosine similarity reduces to $n_{12}/\sqrt{n_1 n_2}$, a variant of the Jaccard similarity.

2 Experiments

2.1 Data

We extracted all complete mouse and human protein sequences from SwissProt Version 44 [23], released in 09/2004 (`http://us.expasy.org/sprot/`), yielding 18,198 protein sequences. We focused on vertebrate data because the multi-domain families that challenge traditional homology identification methods tend to be larger and more complex in vertebrates.

We obtained protein domain architectures from CDART [18]. Among 18,198 sequences, 15,826 sequences have detectable domain architectures. An all-against-all comparison of all 15,826 domain architectures yielded 2,371,608 pairs which share at least one domain.

2.2 Family Identification

To test the various methods we required a set of pairs of proteins with known homology. We constructed a list of sequences from fifteen known families using reports from nomenclature committees (`http://www.gene.ucl.ac.uk/nomenclature/`) and articles from the literature. Types of evidence presented in these articles include intron/exon structure, phylogenetics and synteny. As a result of this process, we have a list of 1,137 proteins each known to be from one of the following families: ADAM (a family of proteins containing A Disintegrin And Metalloproteinase domain) [24,25,26], DVL (Dishevelled protein family) [27,28], FOX (Forkhead transcription factor family) [29,30], GATA (GATA binding proteins) [31,32], Kinase [33,34,35,36], Kinesin [37,38,39], KIR (Killer cell Immunoglobulin-like Receptors) [40,41], Laminin [42,43], Myosin [44,45,46], Notch [47,48,49], PDE (Phosphodiesterases) [50], SEMA (Semaphorin) [51,52], TNFR (Tumor Necrosis Factor Receptors) [53, 54], TRAF (Tumor necrosis factor Receptor Associated Factors) [55], and USP (Ubiquitin Specific Proteases) [56,57].

From this database of sequence families, we constructed a list of positive and negative examples of homologous sequences. For each family, all pairs of sequences with both members in the family are positive examples of homologous pairs, and all pairs sharing at least one domain but with only one member in the family are examples of non-homologous pairs. The number of homologous pairs for the fifteen families studied ranged from 75 pairs in the KIR family to 1,074,570 pairs in the Kinase family. The number of non-homologous pairs ranged from 78 in the FOX family to 184,107 in the Kinase family.

2.3 Performance Evaluation

Our goal is to assign a similarity score to each pair such that all pairs within the family have scores that are higher than scores between protein pairs with one member in the family and one non-family member. We attempted this using the number of common domains similarity score (from here on referred to as the unweighted score) and the weighted number of domains in common similarity score,

trying distinct partner weighting as well as idf weighting and tf-idf weighting. We considered each scoring system both with and without domain-count correction. To evaluate how well these methods could perform in a situation where family identification is not known and therefore a universal cutoff must be used, we also applied these methods to the aggregate set, in which the homologous and non-homologous pairs from all the families are combined to a single set of positive and negative examples.

We evaluated classifier performance using Area Under receiver operating characteristic Curve (AUC) scores. AUC score provides a single measure of classification accuracy, corresponding to the fraction of correctly classified entities given the best possible choice of threshold [58]. The range of AUC scores is from 0.0 to 1. The higher the AUC score, the better the performance. When AUC is 1 the classifier has perfect performance and when AUC is 0.5 the classifier cannot distinguish homologous pairs from non-homologous pairs, *i.e.* the score distributions for positive and negative examples are not separable. When the AUC score is less than 0.5 the positive and negative examples are somewhat separable, but the classifier is separating them in the wrong direction. For this range of AUC scores the classifier is under-performing randomly guessing if the pair is homologous or non-homologous.

3 Results and Discussion

We tested the ability of domain content similarity to classify homologs using the unweighted score, distinct partner weighting, idf weighting, and tf-idf weighting. The results are shown in Table 1. For each method, results are reported both with and without the cosine similarity for domain-count correction. For the unweighted scoring method, Jaccard similarity was also tested, although the results are not reported here as they behave similarly to the cosine similarity results. To understand the effect of domain copy number, we also tested a similarity score based on types of domains in common that ignores domain copy number, but results are not shown as for most families it was similar to the unweighted method.

As the ability to separate positive and negative examples in the aggregate dataset best mirrors the way that these comparison methods will be used in practice, we use these results to judge the overall performance of each comparison method. Because of the large size of the Kinase family, the aggregate results are dominated by the results on Kinase pairs. We therefore consider the aggregate set not including the Kinase family. We see that the unweighted method with no domain-count correction performs poorly with an AUC score of only 0.63. Methods that weighted domains to account for domain promiscuity significantly improve the performance. The aggregate AUC score increases to 0.91 or better by using any of the weighting methods discussed here. Among the three weighting methods considered, we obtain the best performance from distinct partner weighting. It best reflects the evolutionary information encoded by each domain. Cosine similarity provides a domain-count correction and boosts the

Table 1. AUC scores for identifying homologs from different families. Results are shown both with and without domain-count correction using cosine similarity. The last two rows contain the AUC scores for the aggregate system containing all the families, and the aggregate system containing all the families except for Kinase.

	no weighting		distinct partner		idf		tf-idf	
		cosine		cosine		cosine		cosine
ADAM	0.94	0.96	0.98	0.98	0.97	0.98	0.84	0.88
DVL	1.00	1.00	1.00	0.99	1.00	1.00	0.99	1.00
FOX	0.50	1.00	1.00	1.00	0.00	0.99	0.99	0.99
GATA	1.00	0.99	1.00	0.94	0.63	0.87	0.83	0.87
Kinase	0.44	0.71	0.16	0.49	0.09	0.53	0.47	0.46
Kinesin	0.52	0.90	0.99	0.96	0.85	0.93	0.94	0.93
KIR	0.50	0.86	0.50	0.86	0.50	0.77	0.77	0.77
Laminin	0.99	0.97	0.99	0.95	0.97	0.90	0.96	0.97
Myosin	0.64	0.56	0.94	0.65	0.98	0.74	0.76	0.70
Notch	1.00	1.00	1.00	1.00	1.00	1.00	0.96	1.00
PDE	0.54	0.84	0.84	0.87	0.84	0.82	0.82	0.62
SEMA	0.78	0.89	0.99	1.00	0.99	0.99	0.97	0.87
TNFR	0.57	0.96	0.83	0.95	0.30	0.94	0.96	0.95
TRAF	0.85	0.98	0.99	0.99	0.99	1.00	0.99	0.99
USP	0.55	0.84	0.82	0.77	0.57	0.82	0.87	0.85
all	0.46	0.73	0.19	0.55	0.13	0.60	0.51	0.52
all (no Kinase)	0.63	0.89	0.97	0.92	0.91	0.91	0.91	0.88

performance of the unweighted similarity score to 0.89. While there is a significant benefit to using either weighting or a domain-count correction, there is little additional benefit to applying both. The best AUC score of 0.97 is obtained by distinct partner weighting with no domain-count correction. For the Kinase family, the methods based on domain weighting give similar or significantly worse AUC scores as compared to the unweighted methods. The best AUC score for the Kinase family is 0.71, obtained by the unweighted cosine similarity method.

When families are considered individually, the results show that for most families both weighting and domain-count correction are helpful, consistent with the results observed in the aggregate dataset. However, for some families, the performance under different comparison methods differs significantly from the trends of the aggregate dataset. To better understand what information is most informative for domain architecture comparison and the advantages and disadvantages of each method, we will analyze these families in details.

GATA: All methods perform well on the GATA family (AUC score greater then 0.80) except idf weighting, where an AUC score of only 0.63 is obtained. Unlike other scoring systems presented in Table 1, idf weighting does not reflect domain copy number in the similarity score. The poor performance of the idf weighting combined with the good performance of the other scoring systems implies that the domain copy number is important for GATA classification. Each member of the family has two GATA domains, while non-homologous proteins that share a

domain have only one GATA domain. For similarity scores that incorporate the domain copy number, these duplicate domains give GATA family-family pairs improved similarity scores.

To better isolate the effect of domain copy number, we calculated the AUC score where the similarity between two protein sequences is given by the number of shared domain types (ignoring the duplicate copies). For most families, the AUC score using this method was not significantly changed from the unweighted similarity score (which included duplicate domains), but for GATA the AUC score drops from 1.0 to 0.61. The next most significant decline in AUC score occurred in Laminin, where the AUC score dropped from 0.99 to 0.91. For all other protein families the decline in AUC score was less than 0.05.

TNFR: The poor performance (AUC score of 0.57) of the unweighted similarity score in the TNFR family was improved with domain-count correction, distinct partner weighting, or tf-idf weighting. All TNFR family members in our dataset have at least one TNFR domain. The DEATH domain is often also present, and the TNFR_c6 domain is occasionally present, but no member of the TNFR family has more than three domains. TNFR sequences also match unrelated sequences with a large number of domains. Therefore any of the scoring methods that incorporate a domain-count correction (either cosine similarity or tf-idf weighted score) will give a larger similarity score to the homologous TNFR pairs, which have fewer domains.

The distinct partner weighting method works relatively well, as the TNFR domain only has 7 distinct partners, while the DEATH domain has 14 distinct partners. Moreover, the majority of the non-homologous pairs share only a DEATH domain, while the homologous pairs all share a TNFR domain. Conversely, idf weighting fails since the TNFR domain occurs in 47 proteins in the database, while the DEATH domain occurs in only 36. Therefore, the DEATH domain has a larger idf weight than the TNFR domain, so that non-homologous pairs that share a DEATH domain have larger similarity scores than the homologous pairs that share only a TNFR domain. This results in a AUC score of 0.3 for the idf weighted score.

FOX: The FOX family stands out in our test set as the AUC score for idf method is zero. This means that all non-homologous FOX pairs have a higher score than the homologous pairs. Reason behind this is similar to the reason that idf weighting performed poorly with the TNFR family. In our test set, all FOX sequences except one are single domain proteins with the FH domain. The only multi-domain FOX sequence contains a promiscuous domain, the FHA domain. The FH domain is a strong indicator of homology and the FHA domain is not. Although the FH domain exists in a larger number of sequences than the FHA domain (69 vs 27), the FH domain has a smaller number of distinct partners than the FHA domain (FH occurs only with FHA, while FHA occurs with 11 other domains). Therefore, distinct partner gives a higher weight for the FH domain than the FHA domain, while idf gives a higher weight for the FHA domain. Since all homologous pairs share a FH domain and all non-homologous pairs share a

FHA domain, the distinct partner method has a AUC score of 1.0 while the idf weighting method has a AUC score of 0.0.

Kinase: All weighting methods fail for the Kinase family. Surprisingly, the distinct partner weighting and idf weighting score distributions *are* separable, but the distributions of scores are inverted: non-homologous pairs are given larger scores than homologous pairs. This is because the Kinase family is a large, complex family. In our database, there are more than 600 Kinases. Since Kinase proteins, which are characterized by Pkinase domain, have varied domain architectures, the Pkinase domain has the largest number of distinct partners in our dataset. Therefore, although Pkinase domain characterizes the Kinase family, it was given a small score under all weighted scoring systems, and Pkinase is treated like a promiscuous domain. This explains why the score distributions are inverted: Kinase-Kinase pairs share a Pkinase domain which has a low weight, while the non-homologous pairs share a non-Pkinase domain with higher weight.

Myosin: Unlike all other families in our test set domain-count correction does not improve classification performance for the Myosin family. Domain-count correction gives higher similarity scores to domain matches in two proteins with a small number of domains than it does to domain matches between proteins with a larger number of domains. Therefore, when the family members have relatively many domains but share only a few, and the outsider proteins have relatively few domains, then domain-count correction will worsen the AUC score of a family. This occurs in the Myosin family, causing the AUC scores for the comparison methods with domain count correction to be lower than the uncorrected scores.

4 Conclusion

For many families domain architecture comparison is an effective method for identifying homologs. However, these methods are challenged by large families defined by promiscuous domains, such as the Kinase family. For most families, two key aspects are useful for domain content comparisons: a scoring system that corrects for the bias of proteins with a large number of domains and a measure of the importance of the domain in determining family membership. Domain count correction significantly improves the performance for fourteen of the fifteen studied families. Weighting systems such as idf, tf-idf, and distinct partners are helpful for all families in the test set other than the Kinase family. The best performance is obtained by the distinct partner weighting system. For the Kinase family, the weighting systems under study did not work well as they gave a low weight to the Pkinase domain which is a major factor in determining Kinase family membership. For a few families, such as GATA and Laminin, domain copy number useful in determining sequence homology. Outstanding problems that we hope to address in future work include designing a weighting method that works well for the Kinase family as well as the other families and comparing these domain based methods to other homology identification methods based on sequence comparison and alignment coverage.

Acknowledgments

We thank S. H. Bryant, L. Y. Geer, and J. Joseph for helpful discussions. This research was supported by NIH grant 1 K22 HG 02451-01 and a David and Lucille Packard Foundation fellowship.

References

1. Huynen, M.A., Bork, P.: Measuring genome evolution. PNAS **95**(11) (1998) 5849–5856
2. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res **25**(17) (1997) 3389–3402
3. Gilbert, W.: The exon theory of genes. Cold Spring Harb. Symp. Quant. Biol. **52** (1987) 901–905
4. Patthy, L.: Genome evolution and the evolution of exon-shuffling–a review. Gene **238**(1) (1999) 103–114
5. Eichler, E.E.: Recent duplication, domain accretion and the dynamic mutation of the human genome. Trends Genet **17**(11) (2001) 661–669
6. Emanuel, B.S., Shaikh, T.H.: Segmental duplications: an 'expanding' role in genomic instability and disease. Nat Rev Genet **2**(10) (2001) 791–800
7. Kaessmann, H., Zollner, S., Nekrutenko, A., Li, W.H.: Signatures of domain shuffling in the human genome. Genome Res. **12**(11) (2002) 1642–1650
8. Wang, W., Zhang, J., Alvarez, C., Llopart, A., Long, M.: The origin of the jingwei gene and the complex modular structure of its parental gene, yellow emperor, in drosophila melanogaster. Mol Biol Evol **17**(9) (2000) 1294–1301
9. Long, M.: Evolution of novel genes. Curr. Opin. Genet. Dev. **11**(6) (2001) 673–680
10. Long, M., Thornton, K.: Gene duplication and evolution. Science **293**(5535) (2001) 1551
11. Apic, G., Gough, J., Teichmann, S.A.: Domain coalmbinations in archaeal, eubacterial and eukaryotic proteomes. J Mol Biol **310**(2) (2001) 311–325
12. Letunic, I., Goodstadt, L., Dickens, N.J., Doerks, T., Schultz, J., Mott, R., Ciccarelli, F., Copley, R.R., Ponting, C.P., Bork, P.: Recent improvements to the smart domain-based sequence annotation resource. Nucleic Acids Res **30**(1) (2002) 242–244
13. Bateman, A., Birney, E., Cerruti, L., Durbin, R., Etwiller, L., Eddy, S.R., Griffiths-Jones, S., Howe, K.L., Marshall, M., Sonnhammer, E.L.L.: The Pfam protein families database. Nucleic Acids Res **30**(1) (2002) 276–280
14. Corpet, F., Gouzy, J., Kahn, D.: The ProDom database of protein domain families. Nucleic Acids Res **26**(1) (1998) 323–326
15. Gracy, J., Argos, P.: Domo: a new database of aligned protein domains. Trends Biochem Sci **23**(12) (1998) 495–497
16. Heger, A., Holm, L.: Exhaustive enumeration of protein domain families. J Mol Biol **328**(3) (2003) 749–767
17. Murzin, A., Brenner, S., Hubbard, T., Chothia, C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol **247**(4) (1995) 536–40
18. Geer, L.Y., Domrachev, M., Lipman, D.J., Bryant, S.H.: CDART: protein homology by domain architecture. Genome Res. **12**(10) (2002) 1619–1623

19. Bjorklund, A.K., Ekman, D., Light, S., Frey-Skott, J., Elofsson, A.: Domain rearrangements in protein evolution. J Mol Biol **353**(4) (2005) 911–923
20. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management **24**(5) (1988) 513–523
21. Rubin, G.M., Yandell, M.D., Wortman, J.R., Gabor M., G.L., Nelson, C.R., Hariharan, I.K., Fortini, M.E., Li, P.W., Apweiler, R., Fleischmann, W.e.a.: Comparative genomics of the eukaryotes. Science **287**(5461) (2000) 2204–2215
22. Marcotte, E.M., Pellegrini, M., Ng, H.L., Rice, D.W., Yeates, T.O., Eisenberg, D.: Detecting protein function and protein-protein interactions from genome sequences. Science **285**(5428) (1999) 751–753
23. Bairoch, A., Apweiler, R., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale, D.A., O'Donovan, C., Redaschi, N., Yeh, L.S.: The universal protein resource (UniProt). Nucleic Acids Res. **33** (2005) D154–9
24. Nicholson, A.C., Malik, S.B., Logsdon, J.M.J., Van Meir, E.G.: Functional evolution of ADAMTS genes: evidence from analyses of phylogeny and gene organization. BMC Evol Biol **5**(1) (2005) 11
25. Stone, A.L., Kroeger, M., Sang, Q.X.: Structure-function analysis of the adam family of disintegrin-like and metalloproteinase-containing proteins (review). J Protein Chem **18**(4) (1999) 447–465
26. Wolfsberg, T.G., White, J.M.: Adams in fertilization and development. Dev Biol **180**(2) (1996) 389–401
27. Wharton, K.A.: Runnin' with the Dvl: proteins that associate with Dsh/Dvl and their significance to Wnt signal transduction. Dev Biol **253**(1) (2003) 1–17
28. Sheldahl, L.C., Slusarski, D.C., Pandur, P., Miller, J.R., Khl, M., Moon, R.T.: Dishevelled activates Ca2+ flux, PKC, and CamKII in vertebrate embryos. J Cell Biol **161**(4) (2003) 769–77
29. Mazet, F., Yu, J.K., Liberles, D.A., Holland, L.Z., Shimeld, S.M.: Phylogenetic relationships of the fox (forkhead) gene family in the bilateria. Gene **316**(Oct 16) (2003) 79–89
30. Kaestner, K.H., Knochel, W., Martinez, D.E.: Unified nomenclature for the winged helix/forkhead transcription factors. Genes Dev. **14**(2) (2000) 142–146
31. Lowry, J.A., Atchley, W.R.: Molecular evolution of the GATA family of transcription factors: conservation within the DNA-binding domain. J Mol Evol **50**(2) (2000) 103–115
32. Patient, R.K., McGhee, J.D.: The GATA family (vertebrates and invertebrates). Curr Opin Genet Dev **12**(4) (2002) 416–22
33. Robinson, D.R., Wu, Y.M., Lin, S.F.: The protein tyrosine kinase family of the human genome. Oncogene **19**(49) (2000) 5548–5557
34. Hanks, S.K.: Genomic analysis of the eukaryotic protein kinase superfamily: a perspective. Genome Biol **4**(5) (2003) 111
35. Cheek, S., Zhang, H., Grishin, N.V.: Sequence and structure classification of kinases. J Mol Biol **320**(4) (2002) 855–881
36. Shiu, S.H., Li, W.H.: Origins, lineage-specific expansions, and multiple losses of tyrosine kinases in eukaryotes. Mol Biol Evol **21**(5) (2004) 828–840
37. Iwabe, N., Miyata, T.: Kinesin-related genes from diplomonad, sponge, amphioxus, and cyclostomes: divergence pattern of kinesin family and evolution of giardial membrane-bounded organella. Mol Biol Evol **19**(9) (2002) 1524–1533
38. Lawrence, C.J., Dawe, R.K., Christie, K.R., Cleveland, D.W., Dawson, S.C., Endow, S.A., Goldstein, L.S., Goodson, H.V., Hirokawa, N., Howard, J., et. al.: A standardized kinesin nomenclature. J Cell Biol **67**(1) (2004) 19–22

39. Miki, H., Setou, M., Hirokawa, N.: Kinesin superfamily proteins (kifs) in the mouse transcriptome. Genome Res **13**(6B) (2003) 1455–1465
40. Welch, A.Y., Kasahara, M., Spain, L.M.: Identification of the mouse killer immunoglobulin-like receptor-like (Kirl) gene family mapping to chromosome X. Immunogenetics **54**(11) (2003) 782–790
41. Belkin, D., Torkar, M., Chang, C., Barten, R., Tolaini, M., Haude, A., Allen, R., Wilson, M.J., Kioussis, D., Trowsdale, J.: Killer cell Ig-like receptor and leukocyte Ig-like receptor transgenic mice exhibit tissue- and cell-specific transgene expression. J Immunol **171**(6) (2003) 3056–63
42. Engel, J.: Laminins and other strange proteins. Biochemistry **31**(44) (1992) 10643–10651
43. Hutter, H., Vogel, B.E., Plenefisch, J.D., Norris, C.R., Proenca, R.B., Spieth, J., Guo, C., Mastwal, S., Zhu, X., Scheel, J., Hedgecock, E.M.: Conservation and novelty in the evolution of cell adhesion and extracellular matrix genes. Science **287**(5455) (2000) 989–994
44. Richards, T.A., Cavalier-Smith, T.: Myosin domain evolution and the primary divergence of eukaryotes. Nature **436**(7054) (2005) 1113–1118
45. Goodson, H.V., Dawson, S.C.: Multiplying myosins. Proc Natl Acad Sci U S A **103**(10) (2006) 3498–3499 Comment.
46. Foth, B.J., Goedecke, M.C., Soldati, D.: New insights into myosin evolution and classification. Proc Natl Acad Sci U S A **103**(10) (2006) 3681–3686
47. Maine, E.M., Lissemore, J.L., Starmer, W.T.: A phylogenetic analysis of vertebrate and invertebrate notch-related genes. Mol Phylogenet Evol **4**(2) (1995) 139–149
48. Westin, J., Lardelli, M.: Three novel notch genes in zebrafish: implications for vertebrate notch gene evolution and function. Dev Genes Evol **207**(1) (1997) 51–63
49. Kortschak, R.D., Tamme, R., Lardelli, M.: Evolutionary analysis of vertebrate notch genes. Dev Genes Evol **211**(7) (2001) 350–354
50. Degerman, E., Belfrage, P., Manganiello, V.: Structure, localization, and regulation of cGMP-inhibited phosphodiesterase (PDE3). J Biol Chem **272**(11) (1997) 6823–6
51. Raper, J.: Semaphorins and their receptors in vertebrates and invertebrates. Curr Opin Neurobiol **10**(1) (2000) 88–94
52. Yazdani, U., Terman, J.R.: The semaphorins. Genome Biol **7**(3) (2006) 211
53. Locksley, R.M., Killeen, N., Lenardo, M.J.: The tnf and tnf receptor superfamilies: integrating mammalian biology. Cell **104**(4) (2001) 487–501
54. MacEwan, D.J.: TNF ligands and receptors–a matter of life and death. Br. J. Pharmacol. **135**(4) (2002) 855–875
55. Inoue, J., Ishida, T., Tsukamoto, N., Kobayashi, N., Naito, A., Azuma, S., Yamamoto, T.: Tumor necrosis factor receptor-associated factor (TRAF) family: adapter proteins that mediate cytokine signaling. Exp. Cell Res. **254**(1) (2000) 14–24
56. Wing, S.S.: Deubiquitinating enzymes–the importance of driving in reverse along the ubiquitin-proteasome pathway. Int J Biochem Cell Biol **35**(5) (2003) 590–605
57. Kim, J.H., Park, K.C., Chung, S.S., Bang, O., Chung, C.H.: Deubiquitinating enzymes as cellular regulators. J Biochem (Tokyo) **134**(1) (2003) 9–18
58. DeLong, E.R., DeLong, D.M.: Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics **44** (1988) 837–845

Inferring Positional Homologs with Common Intervals of Sequences*

Guillaume Blin[1], Annie Chateau[2,3], Cedric Chauve[2,3,4], and Yannick Gingras[3]

[1] IGM-LabInfo - UMR CNRS 8049, Université Marne-la-Vallée
5 bd. Descartes 77454 Marne-la-Vallée Cedex 2, France
gblin@univ-mlv.fr
[2] LaCIM, Université du Québec À Montréal
CP 8888, Succ. Centre-Ville, H3C 3P8, Montréal (QC), Canada
{chateau, chauve}@lacim.uqam.ca
[3] CGL, Université du Québec À Montréal
ygingras@ygingras.net
[4] Department of Mathematics, Simon Fraser University
8888 University Drive, V5A 1S6, Burnaby (BC), Canada

Abstract. Inferring orthologous and paralogous genes is an important problem in whole genomes comparisons, both for functional or evolutionary studies. In this paper, we introduce a new approach for inferring candidate pairs of orthologous genes between genomes, also called positional homologs, based on the conservation of the genomic context. We consider genomes represented by their gene order – i.e. sequences of signed integers – and common intervals of these sequences as the anchors of the final gene matching. We show that the natural combinatorial problem of computing a maximal cover of the two genomes using the minimum number of common intervals is NP-complete and we give a simple heuristic for this problem. We illustrate the effectiveness of this first approach using common intervals of sequences on two datasets, respectively 8 γ-proteobacterial genomes and the human and mouse whole genomes.

1 Introduction

In the comparison of two genomes, a first natural task is to compare the sequences of their genes (nucleotides or amino-acids) in order to identify homologous genes, that are pairs of genes whose sequence similarity is strong enough to suggest a common ancestral gene. However, due to the evolutionary mechanisms that shape genomes – rearrangements, duplications, both of genomic segments or of whole genomes, gene losses or lateral transfers – homologous relations are often ambiguous and do not induce clear one-to-one relationships between genes of the two genomes [8]. Instead, non-trivial gene families, occurring in several positions in one or both genomes, make difficult to distinguish pairs of orthologous genes. Several methods have been proposed that use the genomic context to distinguish

* Work supported by grants from Génome Québec, NSERC and Coopération Franco-Québecoise.

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 24–38, 2006.

pairs of genes, called *positional homologs*, that are good candidates to be pairs of orthologous genes [8]. In this paper we describe a new approach using the genomic context, based on the notion of *common intervals of two sequences*.

There are two main approaches for inferring positional homologs using the genomic context. In the *exemplar* approach, introduced by Sankoff [21], for every non-trivial gene family, all but one copy in each genome are deleted. The pair of genes that is conserved for each family is called a pair of *ancestral homologs*. The *gene matching* approach is more general as it allows to conserve more than one copy of a gene family and seeks for an unambiguous one-to-one matching between these copies [12]. Both approaches have in common that they lead to represent the two compared genomes by two permutations. These permutations can then be used in several contexts: computing a genomic distance [23], phylogenetic reconstruction [5] or proposing candidate pairs of orthologous genes [12,14].

In both families of methods – exemplar and gene matching –, several combinatorial criteria have been considered in order to clear ambiguous homology relations between genes of the same family. A classical approach is to try to find the resulting pair of permutations that will minimize a given genomic distance between them, like the reversal distance [12,21,23,24] or the reversal and translocation distance [14]. Another approach looks for the pair of permutations that maximizes the conservation of some combinatorial structures in the two resulting permutations, like adjacencies [5] or common intervals of permutations [6]. However, it is important to note that, up to date, all these problems have been shown to be NP-hard [3,7,11]

In the present work, we are interested in computing a gene matching with a method that is inspired by algorithms for global alignment of long genomic sequences. Indeed, given two genomes represented by two sequences of integers – where each integer labels a gene family – a gene matching is nothing else than a global alignment of these two sequences. Here we propose to use the occurrences of common intervals of these two sequences, which are segments having the same gene content with no constraint on gene order or multiplicity, as *anchors* for the final gene matching. Then we recursively match common intervals, using a simple heuristic for the Minimum Inextensible Box Covering problem – which is a NP-hard problem (see Appendix) – until all possible pairs of positional homologs have been chosen. Hence, one of the originality of our method is that it does not try to compute the pair of permutations that is optimal for a given combinatorial criterion, but rely on a greedy approach of recursively matching genomic segments having a common combinatorial structure.

Our method can be seen as an extension of a previous gene matching algorithm, that iteratively matches longest common substrings (LCS) of the two genomes – such LCS representing perfect colinear segments of genes – [23,5], but using a less constrained notion of conserved structure between genomes (common intervals vs. LCS). Common intervals of sequences are interesting with respect to LCS for two reasons. From the biological point of view, they are more adapted to detect sequences of genes that evolved from a common ancestral sequence with events like segmental or tandem duplications, or local rearrangements. From the

algorithmical point of view, the set of all common intervals of two sequences can be computed in quadratic time [22]. Note however that gene deletions or insertions, that are natural evolutionary events, can destroy common intervals, and we discuss this issue in Section 4.

In Section 2, we describe precisely our method to compute a gene matching. In Section 3, we present two experimental studies of our method. We first consider 8 genomes of γ-proteobacteria, and we compare the results of our method with the LCS method of [5]; this experiment raises interesting facts about the properties of these two notions of conserved structures in the comparison of more than two genomes. Next, we consider the human and mouse genomes and we compare our results with the results obtained with the MSOAR method that tries to find the most parsimonious pairs of permutations for the reversals and translocations distance [14]. In Section 4, we describe several ways to improve our initial approach.

2 The New Method

A *matching* between two sequences of integers s_1 and s_2, where each of these integers represents a family of genetic markers – genes here –, is a one-to-one mapping between a subset of the first sequence and a subset of the second sequence, such that pairs of mapped markers belong to the same family [5]. In the method we propose, we compute a matching in four main steps. First, we define candidate anchors for the alignment between the genomes using common intervals of sequences. We briefly recall in the next paragraph some definitions and properties of common intervals of sequences. Then, we exclude anchors that do not exhibit enough structure to produce an accurate matching, using some reasonable selection criteria. Finally we extract from the remaining anchors a consistent subset, and apply the method recursively in each anchor, until we are left only with anchors containing markers of a single family, then we match markers in these boxes. The final result is a matching between the two considered sequences that describe one-to-one correspondences between the genes of the two corresponding genomes.

2.1 First Step: Finding the Anchors

Given a sequence of integers A representing a genome, the alphabet Σ of its gene content, and a subset S of Σ, we define a *location* of S in A as a substring of A that is a word on S. Hence, from a genomic point of view, a location of S in A represents a contiguous region in A with gene content exactly S. The location is *maximal* if this substring cannot be extended on the right or on the left, meaning that the contiguous characters are not in S. A *factor* between two genomes is a subset of Σ which has at least one location in each genome. Note that, for a given factor, there can be several locations in the same genome.

In the following, a *box* will refer to a set of two maximal locations, one in each genome, for a factor S, that is called the *alphabet* of the box[1]. A box is said to

[1] Box are called *common intervals* in [22].

be *trivial* if its alphabet has cardinality 1. We represent a box by the quadruplet of the coordinates of the two corresponding locations.

Example: $S = \{7, 8, 9, 10, 11\}$ is a factor between A and B, with one location in A and two in B, which gives two boxes for S, $(5, 10, 4, 8)$ and $(5, 10, 12, 16)$.

Genome A: 1 2 3 12 [11 8 7 9 9 10] 4 3 1 1 5
Genome B: 6 1 4 [8 9 10 11 7] 2 13 13 [7 8 9 10 11]

The first step of our method consists in the computation of all the boxes between the two considered genomes, that we use as the basic structures to define the final matching. To compute the set of all possible boxes, we use the quadratic-time algorithm of Schmidt and Stoye [22].

2.2 Second Step: Filtering the Set of Boxes

Our experiments with the LCS method showed that a significant number of false positives matched gene pairs (roughly speaking, pairs of matched genes that contain two different genes according to gene names in the Uniprot database, see Section 3 for more details) was removed when LCS of short length were discarded from the analysis. This suggests that subsequences of the two genomes that do not exhibit a strong common combinatorial structure (here measured in terms of the length of the LCS) are more likely to produce wrong pairs of matched genes.

This is why we decided to introduce a similar feature in our method, that discards putative anchors, here boxes, that do not exhibit enough combinatorial structure. For LCS, the natural parameter that defines such a structure is the length, due to the conservation of the order in a LCS. However this is not the case with boxes, due to the less constrained structure that defines them, an issue that we discuss in Section 4.

In this first investigation of using common intervals of sequences that we present in this work, we chose to use a simple geometrical criterion to exclude boxes, that is the length of the smaller location for a given box, called the *minimum side* of the box. The intuition for this choice is that this parameter extends naturally the criterion of length that we used with the LCS method. This filtering step, with the simple criterion we consider, can be done in linear-time in the number of boxes.

We show in Section 3 some experiments using a dataset of 8 bacterial genomes that illustrate the impact of this filtering step on the resulting matchings.

2.3 Third Step: Extracting a Consistent Subset of Candidate Boxes

The third step of the method we propose consists in the computation of the gene matching by a recursive process, that takes as inputs the boxes that were not discarded during the second step.

Before describing this process, we define some combinatorial notions about sets of boxes:

- Intuitively, a box (x_1, x_2, y_1, y_2) defines a rectangle in the 2D plane, defined by the points $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$.
- Two boxes are said to be *compatible* if there exists no vertical or horizontal line that can cross both boxes at once. Formally, $B = (x_1, x_2, y_1, y_2)$ and $B' = (x'_1, x'_2, y'_1, y'_2)$ are compatible if $[x_1, x_2] \bigcap [x'_1, x'_2] = \emptyset$ and $[y_1, y_2] \bigcap [y'_1, y'_2] = \emptyset$.
- A box B id said to be *enclosed* in a box B' if the rectangle B is completely included in the rectangle B'. Formally, $B = (x_1, x_2, y_1, y_2)$ is enclosed in $B' = (x'_1, x'_2, y'_1, y'_2)$ if $[x_1, x_2] \subseteq [x'_1, x'_2]$ and $[y_1, y_2] \subseteq [y'_1, y'_2]$.
- Given a set $\mathcal{B}\{B_1, \ldots, B_n\}$ of boxes, a subset $\mathcal{B}'$ of $\mathcal{B}$ of boxes that are pairwise compatible is said to be *inextensible* if every box of $\mathcal{B}$ that does not belong to $\mathcal{B}'$ is not compatible with at least one box of $\mathcal{B}'$.

The principle of this third step is to select a subset $\mathcal{B}'$ of boxes that is inextensible and of minimum cardinality with respect to this property of being inextensible, then to recursively repeat this process inside each box of $\mathcal{B}'$.

To extract the set $\mathcal{B}'$ of boxes, we consider the Minimum Inextensible Box Covering (MIBC) optimization problem: given a set of n boxes $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$, find a subset$\mathcal{B}' \subseteq \mathcal{B}$ of minimum cardinality such that (1) any pair (B_i, B_j) of boxes of $\mathcal{B}'$ is a pair of compatible boxes, and (2) $\mathcal{B}'$ is inextensible. Note that this problem is a variant of the Maximum Common String Partition problem (MCSP) that occurs naturally in computing a gene matching that minimizes the number of breakpoints in the resulting permutations [3,12]. This is the main reason that led us to introducing this problem for computing gene matchings, as we think it is a very natural way to compute recursively a gene matching from a set of boxes. However, the problem MCSP is NP-hard [16,3] and it is then not surprising that the same result holds for the problem MIBC (the proof is given in Appendix).

Theorem 1. *The* Minimum Inextensible Box Covering *problem is* NP*-hard.*

This hardness result leads us to develop a simple greedy heuristic, inspired from the MIBC problem, that selects an inextensible set of pairwise compatible boxes, by iteratively selecting the box with maximal area.

```
MIBC_heuristic(Input: set of boxes B)
   Create an empty set of boxes B'
   While (B is not empty)
      Select the biggest box Bi, in terms of area, in B
      Add Bi to B'
      Remove every box Bj of B which is incompatible with Bi
   Return B'
```

This heuristic procedure is used recursively until only trivial boxes are left.

```
MIBC_main(Input: set of boxes B)
   Let M=MIBC(B)
   While (M contains at least one non-trivial box)
      For (every non trivial box Bi of M)
```

```
        Remove Bi from M
        Let B' be the set of all boxes of B enclosed in Bi
        Add MIBC(B') to M
Return M
```

The time complexity of this step is polynomial in the number of boxes given in input, which is itself quadratic in the size of the two considered genomes.

2.4 Fourth Step: Matching Genes in Trivial Boxes

Once the third step is completed, we obtain a sequence of trivial boxes, that is boxes of alphabet of size 1, that forms the core of the final matching. Indeed, for all squared trivial boxes of side length 1, that are boxes of area 1, we add the pair of corresponding genes to the final matching.

The case of boxes such that more than two genes are involved is more problematic, as there are several ways to match the genes of such boxes. In the present work, we used the naive strategy that matches genes in increasing order of their positions in the two sequences representing the two genomes. We discuss briefly in Section 3 the influence of this strategy and, in Section 4, we describe several strategies to improve final matching inside the trivial boxes.

3 Experimental Results

We implemented our method in a software called CIGAL (Common Intervals Global ALigner), and we now discuss two experimental studies on two datasets: first 8 genomes of γ-proteobacteria, then the human and mouse genomes.

3.1 Bacterial Genomes

We considered the genomes of the following organisms, that span a wide spectrum in the phylogeny of γ-proteobacteria [2]:

- *Buchnera aphidicola APS* (GenBank accession number NC_002528),
- *Escherichia coli K12* (NC_000913),
- *Haemophilus influenzae Rd* (NC_000907),
- *Pasteurella multocida Pm70* (NC_002663),
- *Pseudomonas aeruginosa PA01* (NC_002516),
- *Salmonella typhimurium LT2* (NC_003197),
- *Xylella fastidiosa 9a5c* (NC_002488),
- *Yersinia pestis CO_92* (NC_003143).

We computed gene families as described in [5]. For each of the 28 pairs of genomes, we computed 6 different gene matchings: first three matchings using the method described in Section 2, respectively with no filtering (the corresponding matching is denoted CI1), with filtering boxes of minimum side 1 (CI2) and 2 (CI3), then three matchings using the LCS method of [5], respectively with no filtering (LCS1), with filtering LCS of length 1 (LCS2) and 2 (LCS3).

We considered two ways to assess the quality of the obtained gene matchings and compare the two methods:

- To assess the accuracy of the matching between genes, we compared the name of coding gene as given by the database UniProt [1], and we defined a gene pair as a *true positive* if both genes have the same name or synonymous names, a *false positive* if the names are different and an *unknown pair* if one of the two genes is not present in UniProt.
- To assess the internal consistency of both methods, we considered the 28 pairwise matchings between pairs of genomes as a graph, called the *combined matchings graph*, whose vertices are the genes of the 8 genomes and edges are given by the matchings. We then computed the proportion of the connected components of this graph that contain at least two genes of a same genome – we call such component *inconsistent components* and components with at most one gene of each genome *consistent components*. Indeed such a situation can be seen as inconsistent with respect to the goal of inferring positional homologs that are candidates to be orthologous genes.
- Subsequently we considered only consistent components and all pairs of genes belonging to the same components that we classified in true positives, false positives and unknowns pairs as described above.
- Finally we classified consistent components into two categories: *perfect*, if all genes in a component have the same name (all gene pairs are true positives), and *imperfect* if at least two genes have different names (note that we discarded components with genes that are not present in Uniprot). Finally, we studied the distribution of perfect components with respect to their size.

The results are given in Tables 1 and 2.

Table 1. Quality of gene pairs and components

	LCS1	LCS2	LCS3	CI1	CI2	CI3
True positives (TP)	19142	13968	10553	18875	13831	10420
True negatives (FP)	3045	1181	789	3324	1500	1054
Unknown pairs (UP)	14420	6244	3630	14211	6419	3818
Number of components	3439	3850	3606	3539	3408	3480
Consistent components (CC)	2907	3704	3537	3117	3147	3382
Ratio of consistent components	0.85	0.96	0.98	0.88	0.92	0.97
TP in a CC	14954	13635	10729	14954	15909	17180
FP in a CC	821	736	596	1114	1661	2433
UP in a CC	7240	5339	3558	8078	9121	11030
Number of perfect components	1531	1723	1538	1628	1817	1962
Ratio perfect/consistent	0.53	0.47	0.43	0.52	0.58	0.58
Number of imperfect components	252	240	190	320	515	687
Ratio imperfect/consistent	0.09	0.06	0.05	0.1	0.16	0.2

First, it is interesting to notice that the number of true positives gene pairs in consistent components decreases when one reduces the minimal length of matched LCS, which is expected, while it increases when one reduces the

Table 2. Distribution of perfect and imperfect components by size

	LCS1	LCS2	LCS3	CI1	CI2	CI3
Perfect components of size 8	258	138	71	254	259	263
Imperfect components of size 8	19	15	9	29	30	32
Perfect components of size 7	209	155	100	218	227	244
Imperfect components of size 7	34	13	9	37	45	86
Perfect components of size 6	157	145	107	172	185	215
Imperfect components of size 6	32	24	19	46	57	114
Perfect components of size 5	206	269	199	222	251	299
Imperfect components of size 5	32	39	24	44	76	135
Perfect components of size 4	236	290	246	253	344	368
Imperfect components of size 4	65	59	49	77	178	186
Perfect components of size 3	465	726	815	509	551	573
Imperfect components of size 3	70	90	80	87	129	134

minimum side of discarded boxes when using common intervals. In fact considering consistent components allows to recover true edges that were lost when discarding boxes of short side, at the price of more false positives and unknown pairs. However, a preliminary study of these imperfect components showed that a significant number seems to be due to the fact that in some cases, when matching two occurrences of a common interval, one has to deal with multiple occurrences of a same family, a phenomenon that does not happen with LCS as the gene order is conserved. We discuss how to improve our method on this point in Section 4.

In terms of components, and especially of consistent components, that are, from our point of view, more important than single gene pairs, it seems that using common intervals lead to the discovery of more (both in terms of number and of ratio) perfect components, here again at the price of more imperfect components. However, it is interesting to notice an important difference between both methods, in terms of the size of the inferred perfect components. Discarding short LCS lead to the loss of large consistent components, in particular perfect components, which is probably due to the fact that LCS represent perfectly conserved colinear segments, and then long LCS that are present in several genomes are quite rare and significant. Hence discarding short LCS clearly improves the accuracy of the results but the price is that small components dominate in the combined matchings graph. On the other hand using common intervals of sequences allows to find more consistent perfect components of large size – at the price of more imperfect components – due to the less strict constraints imposed on matched segments.

From a more biological point of view, we can notice that there seems to be a set of approximatively 250 genes that form maximal perfect components in the combined matchings graph of these 8 genomes. It would be interesting to study more precisely these genes and to compare them with the set of genes used in the phylogenomics analysis of γ-proteobacteria described in [20], or with the set of essential bacterial genes defined in [15].

3.2 Human and Mouse Genomes

In a second experiment, we considered the human and mouse genomes and we compared the results of using common intervals with the method based on LCS and with the recent MSOAR algorithm [14]. We downloaded the two genomes from the MSOAR website and we used the gene pairs of the MSOAR *hit graph* (see [14, Section 3.1]) and a single-linkage clustering to define gene families. In a subsequent step, we deleted from the two genomes all genes that belonged to a family present in only one genome.

Then, we computed 6 gene matchings using both the common intervals method and the LCS methods. These matchings are denoted CI1, CI2, CI3, LCS1, LCS2 and LCS3 as in the previous experiment. Then we classified pairs of positional homologs as true positives (TP) and false positives (FP)on the base of the gene names in Uniprot. The results are summarized in Table 3 below. The results for MSOAR were computed from the result file available on the MSOAR website.

Table 3. Classification of matched pairs in the gene matchings between human and mouse

	MSOAR	LCS1	LCS2	LCS3	CI1	CI2	CI3
Matched pairs	13218	13380	12386	11764	13301	12737	12394
Number of TP	9214	9227	8865	8491	9186	9008	8792
Ratio of TP	0.7	0.69	0.72	0.72	0.69	0.71	0.71
Number of FP	2240	2357	1921	1749	2327	2071	1996
Ratio of FP	0.17	0.18	0.16	0.15	0.17	0.16	0.16

It appears that here again the LCS method performs better, but does lose information faster when short LCS are filtered in comparison to the common interval approach. Moreover, for the same reasons that we described in the previous section, we expect that improving the matchings of genes belonging to boxes with duplicated genes will improve the accuracy of the method based on common intervals. It would also be interesting to understand the reason why discarding boxes with a short minimum side does not increase the number of gene pairs, that could be due either to some properties of the method we used or to differences in the two considered datasets, both from the combinatorial point of view (8 genomes vs. 2 genomes) or the biological point of view.

4 Future Work

In this section we present several ideas that would improve the quality of the matchings computed from common intervals of sequences. In particular, we believe that the general structure of our method allows very naturally to integrate the scores of the all-against-all sequence comparison used to define gene families.

For example, a preliminary analysis of the false positives in both experiments presented in Section 3 suggests that a significant number of them could be corrected by a less naive strategy for matching genes in trivial boxes in the fourth step of our method. A natural improvement could consist in considering the sequence comparison score between the genes of these trivial boxes, picking only pairs of genes that form a best bi-directional hit.

In the second step, we use a basic filtering criterion which rely on the size of the boxes. But it would be of interest to consider other criteria to select the best candidates. It could be useful, for example, to integrate here again the sequence comparison scores as a measure of quality of boxes. This would be a very natural way to balance the effects of the single-linkage strategy used generally to define gene families. One could also think to more sophisticated measures of "quality" of boxes, based on their combinatorial structure: a trivial box of size 3×3 is not necessarily a better candidate than a box of size 2×2 with an alphabet of size 2. Some examples of criteria, like "nestedness", defining what is a "good gene cluster" for genome comparison, can be found in [19].

In the third step, we considered a natural optimization problem, the MIBC problem, that involves only the geometry of boxes, both in its definition and in the heuristic we proposed. It would be interesting to integrate in this step the notion of quality of the boxes, used in the second step, which would lead to a weighted version of the MIBC problem.

From a conceptual point of view, the method we described can be seen as an extension of the LCS method used in [5], where the constraints on the notion of conserved structure used as anchors have been relaxed to accept rearrangements and paralogs. Previous works have already relaxed the notion of LCS, that correspond from a genomic point of view to perfectly colinear segments, allowing insertion or deletion of genes in colinear segments [17,10]. It would be interesting to try to combine these two models and use as anchors common segments with rearrangements, paralogs, insertions and deletions. Several models exist like common intervals with errors [13] or gene teams, that can be computed in polynomial time in the case of two genomes [18]. However, with such relaxed models, the notion of quality of boxes would become more important in order to avoid to use boxes with a weak signal but good geometrical properties.

When considering a dataset of more than two genomes, it can happen that a phylogenetic tree is known for this dataset. In such a case, it would be interesting to perform our pairwise genomes comparisons according to this tree, as it is classical to do it in multiple sequences alignment. We think that such an approach could significantly reduce the number of imperfect components.

Finally in some cases, it would be interesting to consider that a gene in a genome can be matched with more than one other gene in the second genome. One can think for example to genomes that have undergone one or several whole genome duplication, like the yeasts genomes [9]. We think that common intervals of sequences are a good model to compute such generalized matchings.

5 Conclusion

We presented in this work a first study about using common intervals of sequences for computing positional homologs. The experimental results we obtained are very encouraging, despite using very simple approaches for each of the four steps of our method. In particular, the comparison of 8 bacterial genomes seems to indicate that this approach offers a good basis for the comparison of multiple genomes datasets.

Hence, based on this preliminary study, we believe that common intervals of sequences should be considered as a good model for the comparison of whole genomes, which differs from their initial application as a gene cluster model [22].

Moreover, as we described it in Section 4, the very pragmatic approach we proposed, based on four steps, allows to integrate easily more sophisticated technics, and we are currently developing some of these extensions, that will be available in the first release of CIGAL, our Common Intervals Global ALigner.

Acknowledgments. We thank Z. Fu for providing the hit graph of MSOAR [14].

References

1. A. Bairoch *et al.*. The Universal Protein Resource (UniProt). *Nucleic Acids Res.* 33:D154–D159, 2005.
2. E. Belda, A. Moya, F. J. Silva. Genome rearrangement distances and gene order phylogeny in γ-proteobacteria. *Mol. Biol. Evol.*, 22(6):1456–1467, 2005.
3. G. Blin, C. Chauve, and G. Fertin. The breakpoints distance for signed sequences. In *CompBioNets 2004: Algorithms & computational methods for biochemical and evolutionary networks*, vol. 3 of *Texts in Algorithmics*, p. 3–16. King's Coll. Pub., London, 2004.
4. G. Blin, R. Rizzi. Conserved interval distance computation between non-trivial genomes. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, *LNCS* 3595:22–31. Springer, Berlin, 2005.
5. G. Blin, C. Chauve, G. Fertin. Gene order and phylogenetic reconstruction: application to γ-proteobacteria. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:11–20. Springer, Berlin, 2005.
6. G. Bourque, Y. Yacef, N. El-Mabrouk. Maximizing synteny blocks to identify ancestral homologs. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:21–34. Springer, Berlin, 2005.
7. D. Bryant. The complexity of calculating exemplar distances. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, p. 207–212. Kluwer Acad. Press., Dordrecht, 2000.
8. I. J. Burgetz, S. Shariff, A. Pang, E. R. M. Tillier. Positional homology in bacterial genomes. *Evolutionary Bioinformatics Online*, 2:42–55, 2006.
9. K. P. Byrne, K, H, Wolfe. The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. *Genome Res.*, 15(10):1456–1461, 2005.

10. S. B. Cannon, A. Kozik, B. Chan, R. Michelmore, N. D. Young. DiagHunter and GenoPix2D: programs for genomic comparisons, large-scale homology discovery and visualization. *Genome Biology* 4:R68, 2003.
11. C. Chauve, G. Fertin, R. Rizzi, S. Vialette. Genomes containing duplicates are hard to compare. In *Computational Science - ICCS 2006, 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part II.*, *LNCS*, 3992:783–790. Springer, Berlin, 2006.
12. X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–315, 2005.
13. C. Chauve, Y. Diekmann, S. Hebber, J. Mixtacki, S. Rahmann, J. Stoye. On Common Intervals with Errors Report 2006-02, Technische Fakultät der Universität Bielefeld, Abteilung Informationstechnik. 2006.
14. Z. Fu, X. Chen, V. Vacic, P. Nan, Y. Zhong, J. Tang. A parsimony approach to genome-wide ortholog assignment. In *Research in Computational Molecular Biology, 10th Annual International Conference, RECOMB 2006, Venice, Italy, April 2-5, 2006, Proceedings*, *LNCS/LNBI*, 3909:578–594. Springer, Berlin, 2006.
15. J. I. Glass *et al.*. Essential genes of a minimal bacterium. *Proc. Natl. Acad. Sci. USA*, 103(2):425–430, 2006.
16. A. Goldstein, P. Kolman, J. Zheng. Minimum common string partition problem: hardness and approximations. In *Algorithms and Computation, 15th International Symposium, ISAAC 2004, HongKong, China, December 20-22, 2004, Proceedings*, *LNCS*, 3341:473–484. Springer, Berlin, 2004.
17. B. J. Haas, A. L. Dechler, J. R. Wortman, S. L. Salzberg. DAGchainer: a tool for mining segmental genome duplication and synteny. *Bioinformatics*, 20(18):3643–3646, 2004.
18. X. He, M.H. Goldwasser. Identifying conserved gene clusters in the presence of homology families. *J. Comput. Biol.*, 12(6):638–656, 2005.
19. R. Hoberman, D. Durand. The incompatible desiderata of gene cluster properties. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:73–87. Springer, Berlin, 2005.
20. E. Lerat, V. Daubin, N. A. Moran. From gene trees to organismal phylogeny in prokaryotes: the case of the γ-Proteobacteria. *PLoS Biol.*, 1(1):E19, 2003.
21. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.
22. T. Schmidt, J. Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. In *Combinatorial Pattern Matching, 15th Annual Symposium, CPM 2004, Istanbul,Turkey, July 5-7, 2004, Proceedings*, *LNCS*, 3109:97–108. Springer, Berlin, 2004.
23. K. M. Swenson, M. Marron, J. V Earnest-DeYoung, B. M. E. Moret. Approximating the true evolutionary distance between two genomes. In *Proceedings of the Seventh Workshop on Algorithm Engineering and Experiments and the Second Workshop on Analytic Algorithmics and Combinatorics (ALENEX/ANALCO)*, p. 121–129. SIAM Press, New York, 2005.
24. K. M. Swenson, N. D. Pattengale, B. M. E. Moret. A framework for orthology assignment from gene rearrangement data. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:11–20. Springer, Berlin, 2005.

A Proof of NP-Completeness of MIBC

We consider here the decision versions of the problems MIBC and MCSP. We first state formally the MCSP problem. In the following, given a string S, let $S[i]$ and $S[i..j]$ denote respectively the i^{th} character of S and the substring starting at position i and ending at position j of S. Given two strings S and T, a *common substring* of S and T is defined by a quadruplet (i, j, k, l) such that $S[i, j] = T[k, l]$. Two strings S and T are *balanced* if every letter appears the same number of times in S and T. According to [16], a *partition* of a string S is a sequence $P = (P_1, P_2, \ldots, P_m)$ of strings whose concatenation is equal to S. Let P (resp. Q) be a partition of a string S (resp. T). The pair (P, Q) is a *common partition* of S and T if Q is a permutation of P. Given a partition $P = (P_1, P_2, \ldots, P_m)$ of a string, each string P_i is called a *block* of P. We say that a block $P_i = S[k..l]$ *covers* any substring of $S[k..l]$. Two blocks P_i and P_j are *disjoint* if they do not both cover the same character. A common partition (P, Q) of S and T can be naturally interpreted as a bijective mapping from $P = (P_1, P_2, \ldots, P_m)$ to $Q = (Q_1, Q_2, \ldots, Q_m)$. Given a common partition (P, Q), we note $\pi(i) = j$ if P_i is mapped to Q_j. The NP-hard problem [16] MCSP is the following: given two balanced strings S and T and a positive integer s, find a common partition (P, Q) of S and T with at most s blocks.

For the sake of clarity, we recall here the version of the MIBC problem that we consider below: Given a set of n boxes $B = \{B_1, B_2, \ldots, B_n\}$ and a positive integer s', the problem asks to find a subset $B' \subseteq B$ of cardinality lower than or equal to s', such that (1) given any pair (B_i, B_j) of boxes of B', B_i and B_j are compatible and (2) given any box $B_m \in B$ such that $B_m \notin B'$, $\exists B_i \in B'$ such that B_i and B_m are not compatible (B' is said to be *maximal*).

Clearly, MIBC problem is in NP since given a set B' of boxes, one can check in polynomial-time if (1) any pair of boxes of B' is compatible, (2) B' is maximal and (3) $|B'| \leq s'$. We show that given any instance (S, T, s) of MCSP problem, we can construct in polynomial-time an instance (B, s') of MIBC problem such that there exists a common partition (P, Q) of S and T with at most s blocks iff there exists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to s'.

We detail this construction hereafter. Let (S, T, s) be any instance of MCSP problem. For each common substring (i, j, k, l) of S and T, we create a box (i, j, k, l) in B. This can be done in polynomial-time by the use of a generalized suffix-tree for instance. In order to complete the definition of the MIBC problem instance, we define s': $s' = s$. We denote by *box-construction* any construction of this type. An illustration of a box-construction is given in Figure 1.

We now turn to proving that our construction is a polynomial-time reduction from MCSP problem to MIBC problem.

Lemma 1. *Let (S, T, s) be an instance of* MCSP *problem, and (B, s') an instance of* MIBC *problem obtained by a box-construction from (S, T, s). There exists a common partition (P, Q) of S and T with at most s blocks iff there exists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to s'.*

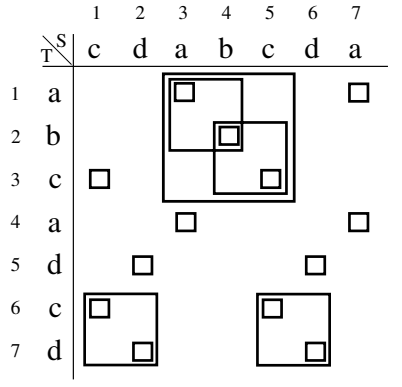


Fig. 1. A schematic view of the set of boxes B obtained by a box-construction of $S = cdabcda$ and $T = abcadcd$. For instance, the largest box is defined by the quadruplet $(3, 5, 1, 3)$.

Proof. ($\Rightarrow$) Suppose we have a common partition (P, Q) of S and T with m blocks ($m \leq s$). We look for a subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to s. We define this set of boxes as follows. For each pair (P_i, Q_j) such that $P_i = S[s_i..e_i]$, $Q_j = T[s_j..e_j]$, $\pi(i) = j$, $1 \leq i, j \leq m$, $1 \leq s_i, s_j, e_i, e_j \leq |S|$, add the box (s_i, e_i, s_j, e_j) to B'.

By construction, notice that the set B' is of cardinality equal to m, which is by definition lower than s. Remains to us to prove that (1) B' is a set of compatible boxes and (2) B' is maximal. By definition, since (P, Q) is a bijective mapping from P to Q, for any $1 \leq i \leq m$ there exists only one $1 \leq j \leq m$ such that $\pi(i) = j$. Moreover, since P (resp. Q) is a partition, the blocks composing P (resp. Q) are disjoint. Therefore, given any box (s_i, e_i, s_j, e_j) of B', there is no box (s_k, e_k, s_l, e_l) in B' such that $[s_i, e_i] \bigcap [s_k, e_k] \neq \emptyset$ or $[s_j, e_j] \bigcap [s_l, e_l] \neq \emptyset$. Thus, B' is a set of compatible boxes.

Let us now prove that B' is maximal. Suppose it is not the case. Thus, there exists a box (s_i, e_i, s_j, e_j) of B that can be added to B' such that B' is still a set of compatible boxes. Then, by construction, the corresponding substrings $S[s_i, e_i]$ and $T[s_j, e_j]$ are not covered by any block of $P \bigcup Q$. Therefore, P and Q are not partitions of S and T; a contradiction. Therefore, if there exists a common partition (P, Q) of S and T with at most s blocks then there exists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to s'.

($\Leftarrow$) Suppose we have a maximal subset of compatible boxes $B' \subseteq B$ of cardinality equal to m ($m \leq s$). We look for two partitions P and Q of S and T each with at most s blocks. We define P and Q as follows. For each box (s_i, e_i, s_j, e_j) of B', we define a block $P_i = S[s_i..e_i]$ and a block $Q_j = T[s_j..e_j]$. Let $P = (P_1, P_2, \ldots, P_m)$ and $Q = (Q_1, Q_2, \ldots, Q_m)$.

By construction, for each box (s_i, e_i, s_j, e_j) of B', there exist a block $P_i = S[s_i..e_i]$ and a block $Q_j = T[s_j..e_j]$. By definition of a box-construction, $S[s_i..e_i]$ and $T[s_j..e_j]$ are common substrings of S and T (*i.e.* $S[s_i..e_i] = T[s_j..e_j]$).

Therefore one can built a bijective mapping of P to Q where for each box (s_i, e_i, s_j, e_j) of B', the corresponding blocks P_i and Q_j are mapped (*i.e.* $\pi(i) = j$). Clearly, Q is thus a permutation of P. Moreover, the partitions P and Q are composed of exactly m blocks which is by definition lower than or equal to s. In order to prove that (P, Q) is a common partition of S and T, it remains to us to prove that P (resp. Q) is a partition of S (resp. T).

First, notice that since B' is a set of compatible boxes, in P (resp. Q) blocks are disjoint two by two. Let us prove that the blocks of P (resp. Q) cover the string S (resp. T). Suppose it is not the case. Since S and T are balanced strings, there exist a position i (resp. j) in S (resp. T) such that $S[i]$ (resp. $T[j]$) is not covered by any block of P (resp. Q) and $S[i] = T[j]$. By definition, since $S[i] = T[j]$, no box in B' covers $S[i]$ and $T[j]$. Moreover, by construction, there exists a box (i, i, j, j) in B which could be added to B' such that B' will still be a set of compatible boxes. Therefore, B' is not maximal; a contradiction. We just prove that if there exists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to s' then there exists a common partition (P, Q) of S and T with at most s blocks. □

On Genome Evolution with Accumulated Change and Innovation*

Damian Wójtowicz and Jerzy Tiuryn

Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
{dami, tiuryn}@mimuw.edu.pl

Abstract. We introduce and analyse a simple discrete probabilistic model of genome evolution. It is based on four fundamental evolutionary events: gene duplication, loss, change and innovation, and it is called *DLCI model*. This is the first such model rigorously analysed. The focus of the paper is around the size distribution of gene families. The formulas for equilibrium gene family sizes are derived showing that they follow a logarithmic distribution. We consider also a disjoint union of DLCI models and we present the result of this study. Some empirical results for microbial genomes are presented.

1 Introduction

Fundamental features of genome evolution are gene duplication and loss [14]. Gene duplication process creates redundancy necessary to free one copy of a gene to evolve a new function and leads to an appearance of paralogous genes. Recall, that any two genes evolved through a duplication from a single ancestral gene, are called *paralogs*. We avoid here a discussion of this important issue of deciding which genes are paralogous. An in depth discussion of this matter can be found in [5]. Here, we assume that all genes have already been clustered into groups of pairwise paralogous genes. We call such groups *gene, or paralog families*. It should be also mentioned that there is no unique or canonical way of clustering. Many different ways of clustering [6,17,23,3,2,21] may yield quite different results.

During the last decade study of the evolution of multigene families has attracted a great deal of attention in comparative genomics. It is not surprising because the paralog families constitute a significant part of a genome, about half of the genes have detectable paralogous gene [18]. New genes to a given lineage, and thus new paralogs families, may emerge as a result of a dramatic gene change after duplication, or via horizontal gene transfer from another species, or by evolution of a protein-coding gene from a non-coding sequence. Genomes contain gene families of various sizes and these sizes change over time. Nevertheless,

* This work was partially supported by the State Committee for Scientific Research (Poland) Grants: 3 T11F 021 28 and 3 TF11 016 28.

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 39–50, 2006.

size distribution of gene families in a genome seems to be invariant over time [17,18,6,7,23,8,16,20,22]. We propose a discrete model of genome evolution in the spirit of Kimura [11], i.e. in the total absence of selective pressure or at least we have to assume that at the genome level selective pressure does not substantially change the shape of gene family size distribution. We are fully aware that such a purely neutralistic model cannot be truly realistic. However, it explains the biological observations and it can be very useful in further discussions in this topic. The essential feature of our model is that it describes the dynamics of the genome at the level of genes. It is based on four fundamental evolutionary events:

- *gene duplication* – an event in which one gene gives rise to two genes which cannot be operationally distinguished between themselves; they remain in the same genome and are therefore paralogs;
- *gene loss* – an event which leads to a removal of a gene from the genome;
- *gene change* – an event (or a cumulative series of events like mutations, rearrangements, recombinations, ...) which lead to such a modification of a sequence that the resulting gene is no longer similar to its parental ancestor and therefore is no longer classified as a paralog;
- *gene innovation* – an event which introduces new genes; it may occur through a horizontal gene transfer between species, especially frequent in case of bacteria or phages [12], or acquired through a series of mutations in a non-coding part of the genome.

The model is called a *DLCI model* (Duplication, Loss, Change and Innovation). We focus on a mathematical analysis of the model from the point of view of paralog family size distribution.

Related Work

A motivation for the present work comes from the study of size distribution of gene families in several microbial genomes which was undertaken in late 90's. Slonimski *et al.* [17], Huynen and van Nimwegen [6] and Jordan *et al.* [7] counted the number of i-element families of paralogous genes (for $i = 1, 2, 3, \ldots$) in several genomes which have been already sequenced. They came up with different claims concerning the shape of the observed distribution: *logarithmic distribution* in [17,7] (the probability of being an i-element cluster is proportional to θ^i/i, where $0 < \theta < 1$), and *power law distribution* in [6] (the probability is proportional to $i^{-\gamma}$, where $\gamma > 1$). It follows from the above contradicting claims that it may be very difficult to decide what actually is the observed distribution if we rely merely on the biological data. A decisive answer should come by adopting a certain mathematical model of a genome evolution together with a rigorous analysis of the distribution within this model. Yanai *et al.* [23] designed a simple model of the genome evolution, but they show only that it is possible to tune the parameters of the model to obtain the distributions that match closely the observed paralog distributions of the genomes considered by the authors. No mathematical analysis was given in this paper.

A few models of genome evolution have already been presented, together with a complete mathematical analysis of the equilibrium frequencies of paralog families [8,9,10,16,19,20,22]. Each of these models is based on gene duplication and loss events, but they differ in other aspects: an absence [19] or a presence of additional evolutionary events like gene change [16,20], innovation [8,9,10,22], or rates of events depending on family size. They also come in two flavours (with respect to time): discrete [19,20,22] or continuous [8,9,10,16,19].

Karev *et al.* show in their papers [8,9,10] that depending on relative rates of birth and death of domains in families (these rates depend on the size of the family and are constant in time) one obtains various equilibrium distributions, including logarithmic and power law. We show in our previous papers that introducing the event of gene change (DLC model [20]) or innovation (DLI model [22]) into the pure DL model with gene duplication and loss [19] (rates of these evolutionary events, excluding the innovation, are constant per gene) results in logarithmic distribution of gene family sizes, while without these features results in geometric distribution. A model with gene change whose rate is constant per family (rates of gene duplication and loss are still constant per gene) is considered by Reed and Hughes in [16]. They show the power law distribution of family sizes in their model.

The main motivation of the paper is to investigate what happens when both features are present in the model. This question was partly investigated in [1], where the genome dynamics with both features was explored by computer simulations. To our knowledge the present paper is the first which addresses the mathematical analysis of asymptotic gene family size distribution in the presence of gene change and innovation.

Main contributions of the paper

The main result of the paper is the statement that the asymptotic size distribution of gene families in the DLCI model is logarithmic, whose parameter depends only on the probability of gene duplication, loss and change, and does not depend on the rate of innovation. A precise formulation of this result is given in Theorem 1. Independence of the asymptotic distribution from the rate of innovation can also be observed for the BDIM model in the main result of [8].

The second result of the paper, Theorem 2, shows that a disjoint union of the DLCI models results in a linear combination of logarithmic distributions. We also give a formula for the weights of this combination. This analysis is motivated by the fact that many gene families evolve at different rates and it is natural to introduce different groups of paralog families, each evolving independently according to its own DLCI model of evolution with individual parameters.

The paper is organised as follows. In Section 2 we describe the DLCI model and present the main result concerning this model. The proof of this result is in Section 3. Study of a disjoint union of DLCI models is presented in Section 4. Section 5 contains a presentation of the experimental results for bacteria genomes. Concluding remarks are presented in Section 6.

2 DLCI Model of Evolution

In this section we describe formally the model of gene duplication, loss, change and innovation in a genome. We view a genome as a finite collection of genes. We treat genes as atomic objects which undergo various evolutionary events. The events happen randomly and each gene evolves independently of other genes. The whole process is discrete in the sense that we observe in discrete time moments the state of the genome. In order to keep track of evolution of paralog families we assume that each gene has its own colour. The intuition behind colours is that two genes have the same colour if, and only if, they are paralogs. Hence monochromatic gene families correspond to families of paralogous genes. We assume that at our disposal we have an unlimited supply of colours. A genome is naturally partitioned into gene families according to colours. A *gene family* in a genome is the set of all genes of the genome which have the same colour. For $i \geq 1$ let $\mathcal{C}_i$ be the collection of all i-element gene families. We call $\mathcal{C}_i$ the i-th *class*. In the present paper we are going to study family size distributions. A *family size distribution* of a genome is a probability distribution $(f_i)_{i\geq 1}$ on positive integers, where $f_i = \frac{|\mathcal{C}_i|}{\sum_{k=1}^{\infty} |\mathcal{C}_k|}$ is the probability of observing an i-element family in the genome.

The DLCI model is parameterised by five non-negative reals: λ, δ, κ, τ (called *evolutionary constants*) and p (called a *probability parameter*) subjected to the condition that $(\lambda + \delta + \kappa)p < 1$. This model consists of two independent subprocesses: internal and external. We describe them separately. One step of genome evolution process is illustrated in Figure 1.

Internal subprocess: Duplication, Loss and Change (DLC) process
This subprocess describes internal changes within the genome (migration of families between classes and formation of new families). In one step of evolution, each gene of genome is independently:

- *duplicated* with probability λp. Both copies of the gene inherit the colour of their parent.
- *lost* with probability δp. The gene is removed from the genome.
- *changed* with probability κp. It changes its colour to a new one, not present in the genome, i.e. the gene starts a new one-element family and is removed from the family to which it belonged.
- remains *unchanged* with probability $1 - (\lambda + \delta + \kappa)p$.

The quantities λ, δ and κ are called *duplication*, *loss* and *change constants*, respectively.

External subprocess: Innovation process
We assume that we have an external source of genes (a black box) which injects genes into the genome randomly according to a certain distribution $(\pi_i)_{i\geq 0}$, whose expected number is τp. The quantity τ is called an *innovation constant*.

It means that in one step the innovation process injects on average τp genes. We assume that the injected genes have brand new colours, i.e. colours which do not occur in the genome, and moreover that the colours of injected genes are pairwise different. It will turn out that the choice of the distribution of the innovation process is irrelevant, as long as the expected number of injected genes is finite and positive. In fact, it will follow from the main result of this paper that the asymptotic gene family size distribution does not even depend on τ. As we will see later innovation process stabilises the size of the genome throughout the evolution.

Thus if we fix the evolutionary constants λ, δ, κ and τ, then we obtain a family of DLCI models for $0 < p < (\lambda + \delta + \kappa)^{-1}$.

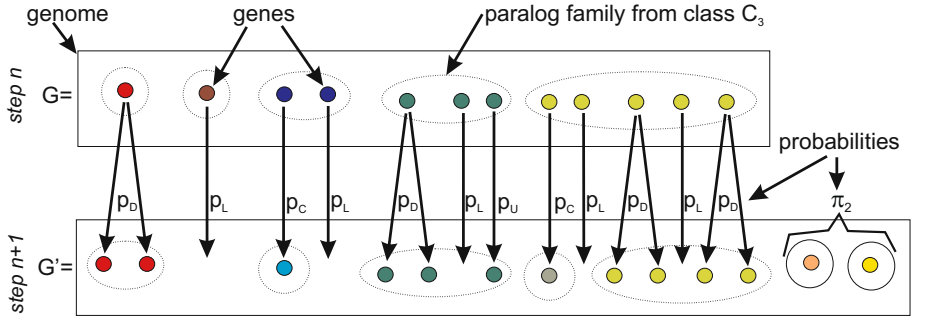


Fig. 1. Illustration of definitions and sample evolution of genome (from G to G'). Here we use the notation $p_D = \lambda p$, $p_L = \delta p$, $p_C = \kappa p$, $p_U = 1 - p_D - p_L - p_C$, and π_2 is a probability of injecting 2 genes into the genome by innovation process.

We assume throughout this paper that the initial genome consists of $K > 0$ one-element gene families. In order to keep the size of the genome finite at the equilibrium we need to make some assumptions on the parameters of the model. Observe that when $\lambda > \delta$ then the size of the genome grows exponentially fast due to the internal subprocess. So, addition of innovation makes things even worse. On the other hand when $\lambda < \delta$, then internal subprocess causes the number of genes decrease exponentially fast, but in this case innovation compensates for this loss. This follows from the following simple computation. Given K genes in the genome, in the next step there are on average $(\delta - \lambda)pK$ genes lost due to internal subprocess and τp genes introduced due to innovation. Hence, on average, the size of the genome in the next step is $\tau p + (1 - \delta p + \lambda p)K$. Repeating this argument n times and passing with n to infinity immediately yields the asymptotic size of the genome equal to $\tau/(\delta - \lambda)$. It is also easy to notice that if $\lambda = \delta$, then the average size of the genome grows only linearly to infinity. Thus we assume that $\lambda < \delta$ to keep the size of genome finite.

Now we can state the main result of our paper.

Theorem 1. *For any positive evolutionary constants λ, δ, κ and τ, such that $\lambda < \delta$, if $p > 0$ is sufficiently small, then for a sufficiently large number of evolution steps the observed family size distribution in the genome is close to the logarithmic distribution with parameter θ, i.e. the probability of observing an i-element family in the genome is close to $C \cdot \theta^i / i$, where $\theta = \lambda/(\delta + \kappa)$, and C is a normalising constant. Moreover the size of the genome is close to $\tau/(\delta - \lambda)$. All the above properties do not depend on the initial size K of the genome, subject to the condition that initially all genes have pairwise different colours.*

3 Sketch of the Proof of Theorem 1

Let $E_i^{(n)}$ be the expected number of i-element families in the above presented process of evolution after n steps ($i \geq 1$, $n \geq 0$). Assuming that the initial genome has K genes with pairwise different colours, we have $E_1^{(0)} = K$ and $E_j^{(0)} = 0$ for $j > 1$. For $n > 0$, we obtain an infinite system of equations for $(E_i^{(n)})_{i \geq 1}$:

$$\begin{cases} E_1^{(n)} = \sum_{j=1}^{\infty} E_j^{(n-1)} \cdot \mathbb{P}(j \rightsquigarrow 1) + \kappa p \sum_{j=1}^{\infty} j \cdot E_j^{(n-1)} + \tau p \\ E_i^{(n)} = \sum_{j=1}^{\infty} E_j^{(n-1)} \cdot \mathbb{P}(j \rightsquigarrow i) \qquad \text{for } i > 1, \end{cases} \tag{1}$$

where $\mathbb{P}(j \rightsquigarrow i) = \sum_{k=0}^{\lfloor i/2 \rfloor} \binom{j}{i-2k,k} (1 - (\lambda + \delta + \kappa)p)^{i-2k} (\lambda p)^k ((\delta + \kappa)p)^{j-i+k}$, for $i, j \geq 1$, is the probability that a gene family of size j gets size i as a result of an internal process of the genome. The situation is slightly different for $i = 1$ since singletons, in addition to the obvious possibility of arriving from other classes by adjusting their size, could have been also created by gene change event or innovation process. The expected number of singletons created by change of genes from one j-element family equals $j \cdot \kappa p$, thus (1) follows.

Let $Q = (\mathbb{P}(j \rightsquigarrow i))_{j,i \geq 1}$, $T = (\tau p, 0, 0, \ldots)$ and $N = (n_{j,i})_{j,i \geq 1}$, where $n_{j,1} = j$ and $n_{j,i} = 0$ for $j \geq 1$ and $i > 1$. System of equations (1) can be rewritten in matrix notation: $(E_1^{(n)}, E_2^{(n)}, E_3^{(n)}, \ldots) = (E_1^{(n-1)}, E_2^{(n-1)}, E_3^{(n-1)}, \ldots)(Q + \kappa p N) + T$. It follows that

$$(E_1^{(n)}, E_2^{(n)}, E_3^{(n)}, \ldots) = (K, 0, 0, \ldots)(Q + \kappa p N)^n + T \sum_{i=0}^{n-1} (Q + \kappa p N)^i \tag{2}$$

for all $n \geq 0$. We are interested in the asymptotic distribution which is derived from (2), when n tends to infinity. Notice that *a priori* it is not clear that such a distribution always exists.

For $i \geq 1$, let $q_{p,i}^{(n)}$ be a probability that a random family in the genome after n steps of evolution process has size i. We keep the parameter p explicit since we will study the distribution when p tends to zero. Then

$$q_{p,i}^{(n)} = \frac{E_i^{(n)}}{\sum_{j=1}^{\infty} E_j^{(n)}}. \tag{3}$$

Of course, for every $n \geq 0$ the distribution $\left(q_{p,i}^{(n)}\right)_{i\geq 1}$ exists. The next result says that the asymptotic distribution exists too.

Proposition 1 (Existence of asymptotic distribution). *Let λ, δ, κ and τ be positive constants such that $\lambda < \delta$, and let $0 < p < (\lambda + \delta + \kappa)^{-1}$. Then there exists the asymptotic distribution of gene family sizes:*

$$(q_{p,i})_{i\geq 1} = \lim_{n\to\infty} \left(q_{p,i}^{(n)}\right)_{i\geq 1}.$$

Sketch of proof: The idea of this proof is similar to the proof presented in our previous paper [19]. It uses generating functions[1]. Thus, let $f_{p,n}(x)$ be a probability generating function for the distribution $\left(q_{p,i}^{(n)}\right)_{i\geq 1}$, i.e. $f_{p,n}(x) = \sum_{i=1}^{\infty} q_{p,i}^{(n)} x^i$. We have to show that the limit function $f_p(x) = \lim_{n\to\infty} f_{p,n}(x)$ exists.

We start with the generating function $d_{p,n}$ for the sequence $E^{(n)} = \left(E_i^{(n)}\right)_{i\geq 1}$, i.e. $d_{p,n}(x) = \sum_{i=1}^{\infty} E_i^{(n)} x^i$. We have shown in [20] (see equation (20)) that if $\tau = 0$, than the generating function for $E^{(n)}$ is $g_{p,n}(x) - g_{p,n}(0)$, where $g_{p,n}(x) = \varphi^{(n)}(x) + \kappa p \gamma^{n-1} \sum_{i=1}^{n-1} \varphi^{(i)}(x)/\gamma^i$, $\gamma = 1 - (\delta - \lambda)p$, $\varphi(x) = (\delta + \kappa)p + (1 - (\lambda + \delta + \kappa)p)x + \lambda p x^2$ and $\varphi^{(i)}$ is i-fold composition of φ with itself (with $\varphi^{(0)}$ being the identity function). Thus, it follows from equation (2) that the generating function for $E^{(n)}$ is $d_{p,n}(x) = h_{p,n}(x) - h_{p,n}(0)$, where $h_{p,n}(x) = K g_{p,n}(x) + \tau p \sum_{i=1}^{n-1} g_{p,n}(x)$. Notice, that $d_{p,n}(1)$ is the number of families in the genome after n steps of the process. Thus we have $f_{p,n}(x) = d_{p,n}(x)/d_{p,n}(1)$.

Next, it follows from Lemma 2 in [20] that for every $|x| < (\delta + \kappa)/\lambda$, there exists the limit $c_p(x) = \lim_{n\to\infty}(d_{p,n}(1) - d_{p,n}(x))$, which is finite. Moreover, for $x \geq 0$ we have: $c_p(x) = 0$ iff $x = 1$. Thus the limit function $f_p(x)$ exists. □

It can be also proved, using Vitali's Theorem, that f_p is an analytic function with radius of convergence $\theta^{-1} = (\delta + \kappa)/\lambda$. Moreover, it satisfies the following functional equation

$$f_p(\varphi(x)) = f_p(x) + \frac{\tau p}{c_p(0)} \frac{\delta + \kappa - \lambda}{\delta - \lambda}(1 - x). \tag{4}$$

See a similar argument in the proof of Theorem 1 in [20]. It should be also clear that $c_p(0) = \lim_{n\to\infty} d_{p,n}(1)$ is the asymptotic number of families in the genome.

It follows from the theory of analytic function (the identity property) that function f_p is the unique analytic function which satisfies (4) and the constraint $f_p(0) = 0$. In this sense, the distribution $(q_{p,i})_{i\geq 1}$ is completely characterised by (4). Unfortunately, it cannot be expressed by elementary functions.

[1] Let $S = (s_i)_{i\in I}$ be a sequence of reals and I be a subset of non-negative integers. A *generating function* for S is a function f defined by power series $f(x) = \sum_{i\in I} s_i x^i$. When S is a probability distribution then the generating function f is called *probability generating function*. See [4].

Now we can conclude the proof of Theorem 1. It follows from equation (4) that

$$\frac{f_p(\varphi(x)) - f_p(x)}{\varphi(x) - x} = \frac{\tau p}{c_p(0)} \frac{\delta + \kappa - \lambda}{\delta - \lambda} \frac{1 - x}{\varphi(x) - x} = \frac{\tau}{c_p(0)} \frac{\delta + \kappa - \lambda}{\delta - \lambda} \frac{1}{\delta + \kappa - \lambda x}.$$

Intuitively it should be clear that the left side of the above equation tends to $f'(x)$, as $p \to 0^+$, because of $\lim_{p \to 0^+} \varphi(x) = x$. Rigorously, it can be explained using similar arguments to those in the proof of Theorem 2 in [20]. It can be also shown (see a similar proof of Lemma 6 in [20]) that there exists the limit

$$c = \lim_{p \to 0^+} c_p(0) = \frac{\tau(\delta + \kappa - \lambda)}{C\lambda(\delta - \lambda)}, \tag{5}$$

where $C = (-\ln(1 - \theta))^{-1}$. The constant c is the asymptotic (with $p \to 0^+$, $n \to \infty$) number of families in the genome. Thus, we get a differential equation $f'(x) = -C/(x - \theta^{-1})$. Solving it, with constraint $f(0) = 0$, we obtain

$$f(x) = -C \cdot \ln(1 - \theta x) = C \cdot \sum_{i=1}^{\infty} \frac{\theta^i}{i} x^i.$$

This completes sketch of the proof of Theorem 1.

4 Disjoint Union of DLCI Models

It is well known that gene families evolve at different rates [13,6], and there is a coherent behaviour of genes from one family, i.e. genes from the same family have the same probabilities of duplication and loss. For example, families that are responsible for life processes of an organism possibly do not show propensity to high change over time. Thus it is natural to assume that we have different groups of paralog families, each group evolving according to a DLCI model with individual parameters of gene duplication, loss, change and innovation.

Let $M > 0$ be a number of different groups of gene families in the above sense. Thus we have a family of M DLCI models with parameters $(\lambda_m, \delta_m, \kappa_m, \tau_m, p_m)_{m=1}^M$, where λ_m, δ_m, κ_m and τ_m are evolutionary constants and p_m is a probability parameter in the m-th group of families. We also assume that initially the m-th group has $K_m > 0$ one-element gene families, for $m = 1, \ldots, M$. Without loss of generality we may assume that $p_m = p$, for all $m = 1, \ldots, M$ (otherwise we can change the values of evolutionary constants). Let us call this model, an *M-DLCI model*.

Let $E_{m,i}^{(n)}$ be the expected number of i-element families in the m-th group of paralog families after n steps of the M-DLCI evolution process. Thus, an expected number of i-element families after n steps of the process equals $E_i^{(n)} = \sum_{m=1}^M E_{m,i}^{(n)}$ and the probability of observing in the genome after n steps an i-element family equals (compare it with equation (3)):

$$q_{p,i}^{(n)} = \frac{\sum_{m=1}^M E_{m,i}^{(n)}}{\sum_{j=1}^{\infty} \sum_{k=1}^M E_{k,j}^{(n)}}. \tag{6}$$

It should be clear that the asymptotic size distribution of paralog families in M-DLC model is a mixture of M logarithmic distributions with parameters $\theta_m = \lambda_m/(\delta_m + \kappa_m)$ (for $m = 1, 2, \ldots, M$). We state it precisely in the following theorem.

Theorem 2. *Let $\lambda_m, \delta_m, \kappa_m, \tau_m > 0$ such that $\lambda_m < \delta_m$ for all $m = 1, \ldots, M$. Then for a sufficiently small value of $p > 0$ and a sufficiently large number of steps of the evolution process, the size distribution of paralog families in M-DLCI model tends to the mixture of logarithmic distributions:*

$$P_{M\text{-}DLCI}(i) \approx \sum_{m=1}^{M} \alpha_m \cdot C_m \frac{\theta_m^i}{i} \qquad i = 1, 2, 3, \ldots,$$

where $\theta_m = \frac{\lambda_m}{\delta_m + \kappa_m}$, $C_m = (-\ln(1-\theta_m))^{-1}$ and $\alpha_m = \frac{c_m}{\sum_{k=1}^{M} c_k}$ is an asymptotic fraction of families in the m-th group among all families (the value of c_m is defined by (7)).

Sketch of proof: We transform equation (6) into $q_{p,i}^{(n)} = \sum_{m=1}^{M} \alpha_{m,p}^{(n)} \cdot q_{m,p,i}^{(n)}$, where $\alpha_{m,p}^{(n)} = \frac{\sum_{j=1}^{\infty} E_{m,j}^{(n)}}{\sum_{k=1}^{M} \sum_{j=1}^{\infty} E_{k,j}^{(n)}}$ and $q_{m,p,i}^{(n)} = \frac{E_{m,i}^{(n)}}{\sum_{j=1}^{\infty} E_{m,j}^{(n)}}$. We have already know, from the analysis of DLCI model (see Theorem 1), that

$$\lim_{p \to 0^+} \lim_{n \to \infty} q_{m,p,i}^{(n)} = C_m \frac{\theta_m^i}{i}.$$

Thus, we have to find $\alpha_m = \lim_{p \to 0^+} \lim_{n \to \infty} \alpha_{m,p}^{(n)}$. It follows from the proof of Proposition 1 that $\lim_{n \to \infty} \sum_{j=1}^{\infty} E_{m,j}^{(n)} = c_{m,p}(0)$. Next using (5) we have

$$c_m = \lim_{p \to 0^+} c_{m,p}(0) = \frac{\tau_m(\delta_m + \kappa_m - \lambda_m)}{C_m \lambda_m (\delta_m - \lambda_m)}. \tag{7}$$

Thus $\alpha_m = \frac{c_m}{\sum_{k=1}^{M} c_k}$, and this completes sketch of the proof. □

5 Experimental Results

In order to compare the observed families of paralogous genes with the values predicted by our model we have examined 17 bacterial genomes, which are listed in Table 1. The paralogous families of these genomes were taken from TIGR-CMR [15] web service[2].

It is known that the distribution of large families of paralogous genes in organisms is very uneven [6,8,17]: large families may span hundreds of classes, most of them empty. For this reason some researchers [17,18] restrict an analysis of families to small classes (cluster size 2 through 6), while others [6,8] group families

[2] `http://www.tigr.org/tigr-scripts/CMR2/paralog_info_form.spl`

into bins, each containing a certain prespecified minimal number of families. In our analysis we choose the latter method. The observed data was fitted to a mixture of two logarithmic distributions $P_{\text{2-DLCI}}(i) = \alpha_1 \cdot C_1\theta_1^i/i + (1-\alpha_1) \cdot C_2\theta_2^i/i$ and the parameters θ_1, θ_2 and α_1 were chosen to minimise the value of Pearson's χ^2–test. For each genome, before the χ^2–test was evaluated we grouped the observed paralog family frequencies into bins, each containing at least 10 families. We put whole classes into bins. As long as the total number of families in the bin is less than 10, we put the next class into it. For 14 out of 17 analysed genomes $P(\chi^2)$ for this model was at least 5%, i.e. no significant difference between the observed and predicted values was detected. The size of the largest family, the values of parameters $\theta_1, \theta_2, \alpha_1$ and the goodness-of-fit $P(\chi^2)$ for analysed genomes are presented in Table 1.

Table 1. Paralogous families in bacterial genomes [15] and the parameters of best-fit 2-DLI model

Genome	max. fam. size	2-DLI model			
		θ_1	θ_2	α_1	$P(\chi^2)$
Bacillus anthracis Ames	107	0.62	0.95	0.76	80 %
Burkholderia mallei	109	0.69	0.97	0.68	25%
Caulobacter crescentus	70	0.37	0.91	0.53	4%
Colwellia psychrerythraea	81	0.66	0.98	0.85	52%
Dehalococcoides ethenogenes	34	0.55	0.96	0.82	47%
Desulfovibrio vulgaris Hildenborough	89	0.64	0.99	0.89	58%
Enterococcus faecalis	87	0.35	0.90	0.53	35%
Geobacter sulfurreduncens	108	0.66	0.97	0.80	5%
Listeria monocytogenes	78	0.42	0.89	0.46	38%
Methylococcus capsulatus Bath	37	0.56	0.92	0.73	74%
Mycobacterium tuberculosis	77	0.32	0.89	0.46	13%
Myxococcus xanthus	236	0.79	0.97	0.78	10%
Pseudomonas fluorescens	140	0.74	0.98	0.77	52%
Pseudomonas putida	110	0.67	0.95	0.65	22%
Pseudomonas syringae	116	0.60	0.97	0.73	1%
Pseudomonas syringae pv phaseolicola	110	0.67	0.97	0.83	4%
Silicibacter pomeroyi	114	0.51	0.93	0.52	13%

It follows from Table 1 that constants θ_1, θ_2 and α_1 for bacterial genomes are grouped around 0.58 ($\pm$0.14), 0.95 ($\pm$0.03) and 0.69 ($\pm$0.14), respectively. Unfortunately, as noticed in [20,22], these constants strictly depend on the method of clustering paralogous genes. Concentration of the value of the mixture weight α_1 around 0.69 indicates that majority of gene families is subject to the first process (with parameter θ_1). Since θ_2 is much bigger than θ_1 and close to 1, it follows that the second group is mostly responsible for large families, that are more evolutionarily conserved.

6 Conclusions

We present in this paper a simple probabilistic model of genome evolution, which includes four types of events: gene duplication, loss, accumulated change and innovation. This is the first model which is based on all these events. Previous models of this process (known in the literature) consider only subsets of these four events. We show that the observed family size distribution in the DLCI model is close to the logarithmic distribution. What is perhaps little unexpected is that this distribution does not depend on the innovation constant and it only depends on relative constants of duplication, loss and change. We propose also a disjoint union of DLCI models and conclude that the resulting distribution is a linear combination of logarithmic distributions.

Presently power law distribution is considered in the literature [6,8,9,10,16,21] as the best fitting distribution for gene family sizes for practically all genomes (a mixture of distributions was not widely considered in this context). We show, taking a reasonable and natural assumption about the existence of groups of families with individual evolutionary rates/constants, that a linear combination of logarithmic distributions can also properly explain the observed data. However, we realize that it is impossible to find statistically significant difference between these two kinds of distributions because the number of families in genomes is too small and there is only a dozen of bins of family classes. Thus we do not claim that our model is the most accurate description of genome evolution, but even this simple model reveals some potentially interesting aspects of this process.

An interesting topic of further research is to investigate the role of changing the rates of evolutionary constants as a function of the family size. In our DLCI model, these rates show a simple proportionality to this size. This is motivated by the fact that large families are more evolutionarily conserved then the small ones, and thus for example it seems that they should have lower rate of gene change. It is obvious that a selective pressure also plays a critical role in the genome evolution and it is still not considered in models in the context of gene family size distribution analysis.

References

1. N.V. Dokholyan, B. Shakhnovich B, E.I. Shakhnovich, *Expanding protein universe and its origin from the biological Big Bang,* Proc Natl Acad Sci USA **99** (2002), 14132–14136.
2. B. Dujon *et al.*, Genome evolution in yeasts. *Nature* 430, pp. 35-44, 2004.
3. A.J. Enright, S. Van Dongen, C.A. Ouzounis, An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research* 30(7), pp. 1575-84, 2002.
4. W. Feller, An introduction to probability theory and its applications. John Wiley and Sons, Inc. New York, London, 1961.
5. W.M. Fitch, Homology, a personal view on some of the problems. *Trends in Genetics*, 16(5), pp. 227-321, 2000.
6. M.A. Huynen, E. van Nimwegen, The Frequency Distribution of Gene Family Size in Complete Genomes. *Molecular Biology Evolution* 15(5), pp. 583–589, 1998.

7. K. Jordan, K.S. Makarova, J.L. Spouge, Y.I. Wolf, E.V. Koonin, Lineage-Specific Gene Expansions in Bacterial and Archeal Genomes. *Genome Research* 11, pp. 555–565, 2001.
8. G.P. Karev, Y.I. Wolf, A.Y. Rzhetsky, F.S. Berezovskaya, E.V. Koonin, Birth and death of protein domains: A simple model of evolution explains power law behavior., *BMC Evolutionary Biology* 2:18, 2002.
9. G.P. Karev, Y.I. Wolf , F.S. Berezovskaya, E.V. Koonin, Gene family evolution: an in-depth theoretical and simulation analysis of non-linear birth-death-innovation models, *BMC Evolutionary Biology* 4/32, 2004.
10. G.P. Karev, Y.I. Wolf , F.S. Berezovskaya, E.V. Koonin, Modelling genome evolution with a diffusion approximation of a birth-and-death process, *Bioinformatics* 21/3, pp. iii12–iii19, 2005.
11. M. Kimura, The Neutral Theory of Molecular Evolution. Cambridge University Press, Cambridge, 1983.
12. Wen-Hsiung Li, Molecular Evolution. Sinauer Associates, Inc., Publishers, Sunderland Massachusetts, 1997.
13. H. Luz, M. Vingron, Family specific rates of protein evolution. *Bioinformatics* 22(10), pp. 1166-1171, 2006.
14. S. Ohno, Evolution by Gene Duplication. Springer Verlag, Berlin, 1970.
15. J.D. Peterson, L.A. Umayam, T.M. Dickinson, E.K. Hickey, O. White The Comprehensive Microbial Resource. *Nucleic Acids Research* 29:1, pp. 123-125, 2001.
16. W.J. Reed, B.D. Hughes, A model explaining the size distribution of gene and protein families, *Math. Biosci.* 189(1), pp. 97–102, 2004.
17. P.P. Slonimski, M.O. Mosse, P. Golik, A. Henaût, Y. Diaz, J.L. Risler, J.P. Comet, J.C. Aude, A. Wozniak, E. Glemet, J.J. Codani, The first laws of genomics. *Microbial and Comparative Genomics* 3:46, 1998.
18. P.P. Slonimski, Comparison of complete genomes: Organisation and evolution. *Proceedings of the Third Annual Conference on Computational Molecular Biology*, RECOMB'99 Stanislaw Ulam Memorial Lecture, ACM Press, 310, 1999.
19. J. Tiuryn, R. Rudnicki, D. Wójtowicz, A case study of genome evolution: from continuous to discrete time model. In Fiala,J., Koubek,V. and Kratochvíl,J. (eds), *Proceedings of Mathematical Foundations of Computer Science 2004*, LNCS 3153, Springer, pp. 1–24, 2004.
20. J. Tiuryn, D. Wójtowicz, R. Rudnicki, A Model of Evolution of Small Paralog Families in Genomes. (submitted for publication), 2006.
21. Y.I. Wolf, N.V. Grishin, E.V. Koonin EV, Estimating the number of protein folds and families from complete genome data, *J. Molecular Biology* 299,pp. 897–905, 2000.
22. D. Wójtowicz, J. Tiuryn, On genome evolution with innovation. *Mathematical Foundations of Computer Science 2006* (to appear).
23. I. Yanai, C.J. Camacho, C. DeLisi, Predictions of Gene Family Distributions in Microbial Genomes: Evolution by Gene Duplication and Modification. *Physical Review Letters* 85(12), pp. 2641–2644, 2000.

Paths and Cycles in Breakpoint Graphs of Random Multichromosomal Genomes

Wei Xu[1], Chunfang Zheng[2], and David Sankoff[1]

[1] Department of Mathematics and Statistics, University of Ottawa, Canada K1N 6N5
[2] Department of Biology, University of Ottawa, Canada K1N 6N5
{wxu060, czhen033, sankoff}@uottawa.ca

Abstract. We study the probability distribution of genomic distance d under the hypothesis of random gene order. We interpret the random order assumption in terms of a stochastic method for constructing the alternating colour cycles in the decomposition of the bicoloured breakpoint graph. For two random genomes of length n and χ chromosomes, we show that the expectation of $n + \chi - d$ is $O(\frac{1}{2}\log\frac{n+\chi}{2\chi} + \frac{3}{2}\chi)$. We then discuss how to extend these analyses to the case where intra- and interchromosomal operations have different probabilities.

1 Introduction

Though there is a large literature on chromosomal rearrangements in genome evolution and algorithms for inferring them from comparative maps, there is a need for ways to statistically validate the results. Are the characteristics of the evolutionary history of two related genomes as inferred from an algorithmic analysis different from the chance patterns obtained from two unrelated genomes? Implicit in this question is the notion that the null hypothesis for genome comparison is provided by two genomes, where the order of markers (genes, segments or other) in one is an appropriately randomized permutation of the order in the other. In a previous paper [5], we formalized this notion for the case of the comparison of two random circular genomes, such as are found in prokaryotes and in eukaryotic organelles. We found that the expected number of inversions necessary to convert one genome into the other is $n-O(\frac{1}{2}\log n)$, where n is the number of segments (or other markers). Related work has been done by R. Friedberg (personal communication) and by Eriksen and Hultman [1].

In another paper [4], we used simulations to throw doubt on whether the order of syntenic blocks on human and mouse retains enough evolutionary signal to distinguish it from the case where the blocks on each chromosome are randomly permuted.

In this paper, we begin to bridge the gap between mathematical analysis of simple genomes and simulation studies of advanced genomes. We extend the mathematical approach in [5] to the more difficult case of genomes with multiple linear chromosomes, such as those of eukaryotic nuclear genomes, which not only undergo inversion of chromosomal segments, but also interchromosomal translocation. The presence of chromosomal endpoints changes the problem

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 51–62, 2006.

in a non-trivial way, requiring new mathematical developments. Key to our approach in this and previous papers is the introduction of randomness into the construction of the breakpoint graph rather than into the genomes themselves, which facilitates the analysis without materially affecting the results. One aspect of this is that the random genomes with multiple linear chromosomes may also include one or more small circular fragments, or *plasmids*.

Our main result is that the number of operations necessary to convert one genome into the other is $n - O(\frac{1}{2}\log\frac{n+\chi}{2\chi} + \frac{3}{2}\chi)$, where χ is the number of chromosomes in each genome. This result is validated by exact calculations of a recurrence up to large values of n and χ, by simulations, by analytic solution of a somewhat relaxed model, and by solving the limiting differential equation derived from the recurrence.

We also propose models where the the randomness is constrained to assure a realistic predominance of inversion over translocation. We use simulations of this model to demonstrate how key properties of the breakpoint graph depend on the proportion of intra- versus interchromosomal exchanges.

2 The Breakpoint Graph: Definitions and Constructions

Genomic distances can be efficiently computed using the bicoloured *breakpoint graph*. In this graph, the $2n$ vertices are the 5' and the 3' ends of each marker, be it a gene, a probe or a chromosomal segment, occurring orthologously in both genomes. The edges represent the *adjacencies* between the ends of successive markers on either DNA strand in the two genomes. We colour the edges from one genome (R) red, the other (B) black. With the addition of dummy vertices (*caps*) at the endpoints of the χ_R and χ_B linear chromosomes, and dummy edges connecting each cap to one marker end, the breakpoint graph decomposes automatically into alternating colour cycles and alternating colour paths, the latter starting and terminating at caps. The number b of breakpoints – marker adjacencies in one genome not occurring in the other genome – and the number κ of cycles and paths in the breakpoint graph are the dominant components in formulae for genomic distance. Indeed, with the slightly generalized notions of breakpoint and cycle in reference [7], the most inclusive formulation of genomic distance, encompassing inversions, reciprocal translocations, chromosome fission and fusion, and block exchange (including transposition), where the latter operation is counted as two steps, the genome distance d satisfies

$$d = b - \kappa. \tag{1}$$

We add red caps to both ends of each chromosome in R and black caps to both ends of each chromosome in B. The breakpoint graph has $2n + 2\chi_R + 2\chi_B$ vertices corresponding to the $2n$ marker ends and the $2\chi_R$ red caps and $2\chi_B$ black caps. The adjacencies in R determine $n - \chi_R$ red edges and the adjacencies in B determine $n - \chi_B$ black edges. The caps adjacent to chromosome ends determine a further $2\chi_R$ red edges and $2\chi_B$ black edges, for a total of $n + \chi_R$ red edges and

$n + \chi_B$ black edges. Because each marker vertex is incident to exactly one red and one black edge, the graph decomposes naturally into $\chi_R + \chi_B$ alternating colour *paths* and one or more disjoint alternating colour *cycles*.

3 The Randomness Hypothesis and the Relaxation of Linearity

The key to a mathematically tractable model of random genomes is to relax the constraint that genomes B and R are composed only of χ_R and χ_B linear chromosomes. The only structure we impose, in each genome separately, is that every cap is adjacent to a non-cap vertex and every vertex is adjacent to one other vertex or to a cap, and that these pairings, which define the black lines from genome B and the red lines from genome R in the breakpoint graph, are constructed at random from the $2n$ vertices and $2\chi_R$ or $2\chi_B$ caps. Since every vertex is incident to exactly one black line and one red line as before, the breakpoint graph still decomposes into disjoint alternating colour paths and cycles. Studying the statistical structure of the set of paths and cycles is facilitated by relaxing the condition that genomes B and R are composed only of χ_R and χ_B linear chromosomes, but the consequence is that the random choice of vertices defines a genome that contains not only this number of linear chromosomes, but also in general several circular plasmids. There are partial mathematical results (Theorem 1.4 in [2]) which strongly suggest that this relaxation does no violence to the probabilistic structure of the breakpoint graph.

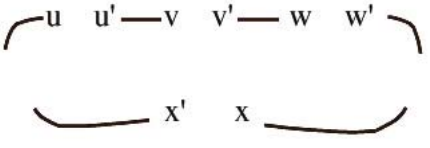


Fig. 1. Random matching gives rise to plasmids (when x and x' are two ends of the same marker) as well as linear chromosomes (when x and x' are two caps.)

For example, consider any vertex v, as in Figure 1. The chromosome containing v in genome R also contains v', the vertex at the other end of the same marker. It also contains u' and w, where u' and v are chosen by the random process to be adjacent in that genome, the vertex w adjacent to v', as well as w and u, and so on. Eventually, the two ends of construction will arrive at the two ends of a single marker, closing the circle, or two red caps, defining a linear chromosome.

Note that these considerations are independent of the properties of the alternating cycle containing v in the breakpoint graph, which would require information about *both* genomes R and B.

4 How Many Paths and Cycles?

In Section 3, we discussed the structure of the individual genomes. We now examine the structure of the breakpoint graph determined by the two genomes.

We will assume $\chi_R = \chi_B = \chi$ to simplify the presentation; the full version of this paper will give the details for the general case of different numbers of chromosomes in R and B.

4.1 The Case of No Caps – Circular Chromosomes

In [5], we pointed out that in the relaxed model for a random genome without caps, the the expected number of cycles approaches

$$\kappa = \log 2 + \frac{\gamma}{2} + \frac{1}{2}\log n, \tag{2}$$

where $\gamma = \lim_{n\to\infty}\left[\sum_{i=1}^{n}\frac{1}{i} - \log n\right] = 0.577\ldots$ is Euler's constant. We also cited partial mathematical results [2] and carried out simulations, both of which indicate that (2) also holds true without the relaxation, i.e. where each genome consists of a *single* DNA circle.

4.2 Linear Chromosomes

Where there are $\chi > 0$ linear chromosomes, so that each genome is assigned 2χ caps, we can take an initial view of the model as a random ordering of $2n$ "colourless" vertices and 2χ coloured vertices, where $2\chi - 1$ of the latter each represent the concatenation of two caps, one at the end of one path and one at the beginning of another path, while one vertex represents the end of the last (2χ-th) path. The vertices ordered after the latter must be on cycles rather than paths, and will be reordered in a later step.

What proportion of the vertices are on each path? As $n \to \infty$, the model becomes simply that of a random uniform distribution of 2χ points (the concatenated two-cap vertices and the final single cap) on the unit interval. The probability density of the distance between two order statistics x_k and x_{k+1}, representing the length of an alternating colour path, is the same for all $0 \le k \le 2\chi - 1$, where $x_0 = 0$:

$$q(x_{k+1} - x_k) = 2\chi(1 - (x_{k+1} - x_k))^{2\chi-1}, \tag{3}$$

with mean $1/(2\chi+1)$ and variance $\chi/[(2\chi+1)^2(\chi+1)]$. The probability density of the last order statistic, representing the sum of the lengths of all 2χ paths, is

$$q_\Sigma(x_{2\chi}) = 2\chi x_{2\chi}^{2\chi-1}, \tag{4}$$

with mean $2\chi/(2\chi+1)$ and variance $\chi/[(2\chi+1)^2(\chi+1)]$.

For the purposes of calculating genomic distance, it is convenient to introduce a certain asymmetry between the red and black genomes. Formula (1) is generally equivalent to

$$d = b + \chi - \kappa - \psi, \tag{5}$$

where ψ is the number of paths having at least one red cap. The explanation for this can be traced in [6] and [7]. Where the caps are distributed randomly, as in our model, the proportion of such paths is $\frac{3}{4}$, and the expected value of ψ is $\frac{3}{2}\chi$.

4.3 Cycles

The proportion of the genomes that is in cycles is just what is left over after the paths are calculated, in the limit $1 - x_{2\chi}$. We ignore the initial linear ordering of the these remaining vertices and instead choose two new random bipartite matchings among them, representing their configurations in genomes R and B. Then, in the limit, the number of cycles that will be constructed from this proportion of the genome is $\kappa_\chi(x_{2\chi}) = \log 2 + \frac{\gamma}{2} + \frac{1}{2}\log[(n+\chi)(1-x_{2\chi})]$, from (2). Thus, from (4), the expectation of of the random variable κ_χ is

$$\begin{aligned}
&\int_0^1 \kappa_\chi(x) q_\Sigma(x) dx \\
&= \int_0^1 \frac{2\chi}{2}\log(1-x)x^{2\chi-1}dx + \log 2 + \frac{\gamma}{2} + \frac{\log(n+\chi)}{2} \\
&= \log 2 + \frac{1}{2}\left[-\sum_{i=1}^{2\chi}\frac{1}{i} + \gamma + \log(n+\chi)\right] \\
&\sim \log 2 + \frac{1}{2}\log\frac{n+\chi}{2\chi}. \qquad (6)
\end{aligned}$$

While we will confirm the second term in (6) in subsequent sections, the $\log 2$ term is likely an extraneous result of our mathematical interpretation of the random linear chromosome model in the previous paragraph, Section 4.2, or the separate passages to the limit forms of κ_χ and q_Σ before integrating.

5 An Exact Recurrence for the Expected Number of Cycles

We build the random breakpoint graph as follows. First we construct R as a random match of the $2n$ labelled vertices and 2χ red caps, under the condition that no caps are matched to each other. Then we construct B as a random match of the $2n$ vertices, using black edges as in Figure 2[1] We will calculate the expected number of cycles completed during this construction by studying what happens as each edge in B is added,

Consider the connected components of the graph at any stage during its construction. They are either cycles, inner edges (paths incident to no cap), cap edges (paths incident to exactly one cap), or completed paths (with caps at either end). Let $N(\kappa, l, m)$ be the number of (equiprobable) ways graphs with κ

[1] We may ignore the black caps and cap edges in this process, since each step consists of joining two red vertices by a black edge.

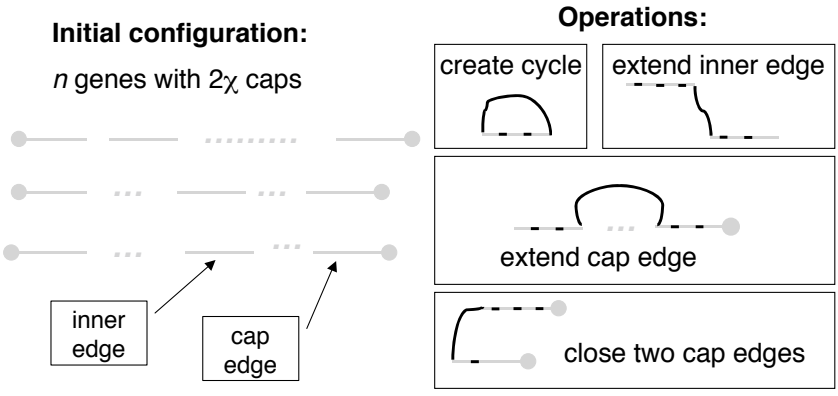


Fig. 2. Initial configuration of edges and caps. Operations of extending inner edges or cap edges and completing cycles or paths.

cycles are produced by this process starting with l inner edges and $2m$ cap edges. Initially $m = \chi$ and $l = n - \chi$, composed entirely of red edges.

There are $\binom{2m+2l}{2}$ ways of adding a black edge:

- The number of cycles can be increased by 1 if the two ends of an inner edge are connected. This decreases l by 1 and may happen in l ways.
- Two inner edges can be connected to form one inner edge. Again l decreases by 1. This can be done in $\binom{2l}{2} - l$ ways.
- One end of an inner edge is connected to a cap end. Again l decreases by 1. This can be done in $4lm$ ways
- Two cap ends are connected. Here $2m$ decreases by 2, and m decreases by 1. This can be done in $\binom{2m}{2}$ ways

Then

$$\begin{aligned} N(\kappa, l, m) = {} & lN(\kappa - 1, l - 1, m) \\ & + (\binom{2l}{2} - l + 4lm)N(\kappa, l - 1, m) \\ & + \binom{2m}{2} N(\kappa, l, m - 1), \end{aligned} \tag{7}$$

where $N(\kappa, 0, m) = 0$ for $\kappa > 0$ and $N(0, 0, m) = (2m)!/2^m$, for all positive m.

The expected number of cycles constructed during our procedure will be

$$\begin{aligned} E[\kappa(l, m)] &= \frac{\sum_\kappa \kappa N(\kappa, l, m)}{\sum_\kappa N(\kappa, l, m)} \\ &= \frac{\sum_\kappa \kappa N(\kappa, l, m)}{\prod_{i=1}^{l+m} \binom{2i}{2}} \end{aligned} \tag{8}$$

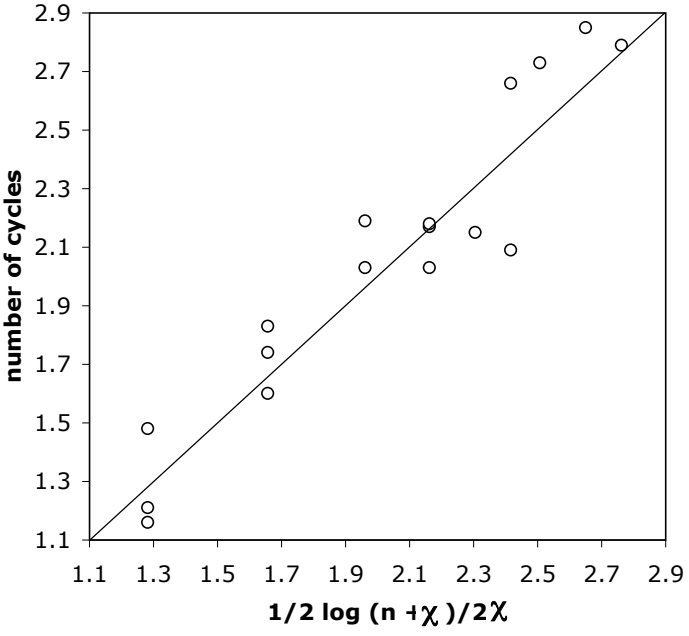


Fig. 3. Simulations for $\chi = 20$ and n ranging from 500 to 10,000. Each point represents the average of 100 pairs of random genomes.

since there are $\prod_{i=1}^{l+m} \binom{2i}{2}$ ways of adding black edges until the number of inner edges and the number of cap edges are both zero.

From (7) and (8), we find that

$$\begin{aligned} E[\kappa(l,m)] = &\frac{2m(2m-1)}{(2l+2m)(2l+2m-1)} E[\kappa(l,m-1)] \\ &+[1 - \frac{2m(2m-1)}{(2l+2m)(2l+2m-1)}] E[\kappa(l-1,m)] \\ &+\frac{2l}{(2l+2m)(2l+2m-1)}. \end{aligned} \tag{9}$$

6 Limiting Behavior of $E[\kappa(n,\chi)]$

Motivated by equation (6), if we calculate $E[\kappa(l,m)]$ for a large range of values of l and m, we find that to a very high degree of precision, the values fit

$$E[\kappa(n,\chi)] = \frac{1}{2} \log \frac{n+\chi}{2\chi}, \tag{10}$$

without the $\log 2$ term in equation (6).

Furthermore, when we simulate 100 pairs of random genomes with 20 chromosomes, for a large range of values of n, using a strictly ordered model rather than the relaxed models in Sections 4-5 above, and count the number of cycles in their breakpoint graphs, the average trend corresponds well to equation (10). This is seen in Figure 3.

Rewriting recurrence (9) for $t(l,m) = E[\kappa(l,m)]$ as

$$\begin{aligned} t(l,m) - t(l-1,m) = & \frac{2m(2m-1)}{(2l+2m)(2l+2m-1)} \\ & \times [t(l,m-1) - t(l-1,m-1) - t(l-1,m) + t(l-1,m-1)] \\ & + \frac{2l}{(2l+2m)(2l+2m-1)}, \end{aligned} \tag{11}$$

suggests that any limiting formulation for t should satisfy the equation:

$$\frac{dt}{dl} = \frac{2m(2m-1)}{(2l+2m)(2l+2m-1)}\left(\frac{dt}{dm} - \frac{dt}{dl}\right) + \frac{2l}{(2l+2m)(2l+2m-1)}. \tag{12}$$

The solutions of (12) are of form $t = \frac{1}{2}\log\frac{l+m}{2m} + C$, for any constant C. This confirms that $E[\kappa]$ is $O(\frac{1}{2}\log\frac{n+\chi}{2\chi})$. Comparison with the boundary condition $\chi = n$ in the discrete model, where each chromosome in our construction starts with two cap edges and no inner edges, so that $\kappa \equiv 0$, further confirms the computationally-established value of $C = 0$.

7 Differential Rates of Inversion and Translocation

The models we have been investigating assume that adjacencies between vertices are randomly established in one genome independently of the process in the other genome. For multichromosomal genomes, this means that the probability that any particular pair of adjacent vertices in the black genome are on the same chromosome in the red genome is of the order of χ^{-1}. This suggests that there are far fewer intrachromosomal exchanges during evolution than interchromosomal, in the approximate ratio of $\chi^{-1} : 1$, which, in the mammalian case, comes to about $0.05 : 1$, a tiny minority. In point of fact, intrachromosomal processes such as inversion represent not a minority, but a substantial majority of evolutionary events. Table 1 gives the estimated ratio of intrachromosomal events to interchromosomal events among six vertebrate species. This ratio depends on the resolution of the syntenic block evidence used to estimate the events; at finer resolutions than the 1 Mb used for the table, the ratio increases considerably. Even for the mouse-dog comparison the ratio is more than 1 at a 300 Kb resolution, while most of the other comparisons have a $2 : 1$ ratio or more.

What is the importance of this tendency for our theoretical analysis? First, there is no direct connection between the number of translocations among pairs of chromosomes and the number of adjacent vertex pairs in one genome that are on different chromosomes in the other, though they are roughly correlated; in

Table 1. Ratio of intrachromosomal events to interchromosomal ones, at a resolution of 1Mb. Calculated from estimates in [3]. Asymmetries due to construction of primary data sets in the UCSC Genome Browser and to asymmetry in the estimator used.

	human	mouse	chimp	rat	dog	chicken
human	\	1.2	-	1.6	1.7	2.9
mouse	1.3	\	1.1	2.3	0.7	1.3
chimp	15	1.4	\	-	-	-
rat	1.5	1.7	-	\	-	-
dog	1.9	0.7	-	-	\	-
chicken	4.5	1.8	-	-	-	\

some examples, a single translocation could "remove" many such edges. Nevertheless, these edges are the only obvious property of the breakpoint graph that we can model. For example, in our derivation of the recurrence (9) in Section 5, we could divide the end vertices of inner edges into χ classes corresponding to the χ chromosomes, as in Figure 4. Then by adjusting the relative probabilities of choosing intra-class edges versus inter-class edges, we can indirectly model differing proportions of inversions versus translocations. The removal of the simplifying assumption of equiprobable edge choice, however, would greatly complicate the analysis leading up to (9) and hence to (10).

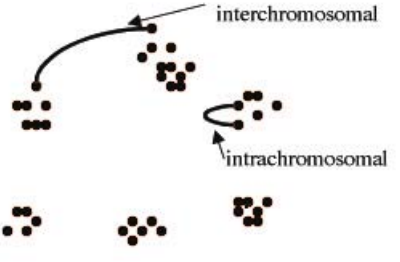


Fig. 4. Partitioning vertices into classes according to chromosomes in genome R. Two kinds of edges with differing probabilities, corresponding roughly to inversion versus translocation rates.

Leaving the theoretical aspects open, then, we propose a simulation approach to the question of the how the inversion-translocation ratio affects the breakpoint graph. For this simulation, our choice of parameters is inspired by the human-mouse comparison with 270 syntenic autosomal blocks at a resolution of 1 Mb. For simplicity we set $\chi = 20$ in both genomes. We know the genomic distance is about 240, but because of breakpoint reuse we need to use 405 operations for the algorithm to infer 240 (and there will obviously be little connection between the operations inferred by the algorithm and the operations actually producing the genomes).

We initialized the simulations with a genome having a distribution of chromosome sizes, in terms of numbers of blocks, patterned roughly after the human

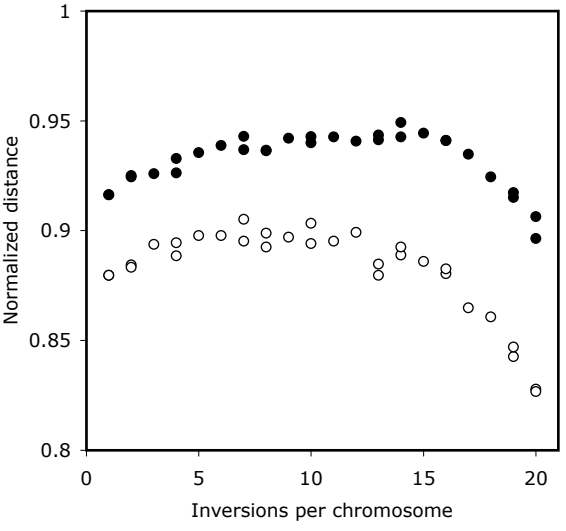


Fig. 5. Effect of changing inversion-translocation proportions. Open dots: before discarding 2-cycles. Filled dots: after discarding 2-cycles.

genome. We then used random inversions and random translocations to produce the second genome. The translocations were conditioned not to result in chromosomes smaller than a certain threshold or larger than a certain cap.

We sampled 10 runs with r inversions per chromosome and $405 - 20r$ translocations, for each $r = 1, \ldots, 20$. In Figure 5, we show that the average inferred distance (normalized by dividing by 270, the number of blocks) rises slowly with the increasing proportion of inversions, then falls precipitously as translocations became very rare. One artifact in this result is due to "two-cycles", representing genes that are adjacent in both genomes. In the breakpoint graphs of random genomes, 2-cycles occur rarely; the expected number of them has a limiting probability of $\frac{1}{2}$. And there are no 2-cycles in breakpoint graphs created from real genome sequence data. (If two syntenic blocks were adjacent and in the same orientation in both genomes, they would simply be amalgamated and treated as a single, larger, block.) Breakpoint graphs created from random inversions and translocations, however, will tend to retain some 2-cycles even after a large number of operations. It takes a very large number of operations before we can be sure that all adjacencies will be disrupted. For the sake of comparability, therefore, we should discard all two cycles and reduce n by a corresponding amount. This done in Figure 5 and it does reduce somewhat the variability of the normalized distance with respect to the inversion-translocation proportion, because the number of 2-cycles rises from about 10 per run when there are few inversions per chromosome, to more than 20 per run when there are 19 or 20 inversions per chromosome, and very few translocations.

Nevertheless, there remains two clear effects, an initial rise in the genomic distance, which will not discuss here, and a larger drop in the distance when

nearly all the operations are inversions. This drop is largely accounted for by an increase in the number of cycles from an average of 2 per run when there are less than 15 inversions per chromosome to 10 cycles per run, when there are 20 inversions per chromosome. To explain this, we observe that insofar as translocations do not interfere, the evolution of the genomes takes place as if each chromosome was evolving independently on is own. But from (10), we could then expect about $\frac{1}{2}\log(\frac{1}{2} + 270/40) \sim 1$ cycle per chromosome or 20 for the whole genome. Were the last five translocations removed from our simulation, we could expect almost that increase in the number of cycles, remembering of course, that with only 405 operations, we are still far from a complete randomization.

8 Discussion

We have continued the development of probabilistic models of random genomes, with a few to testing the statistical significance of genome rearrangement inferences. Here, we have focused on the breakpoint graphs of multichromosomal genomes and found that the expectation of the distance between two random genomes, based on equation (5), is $n + \chi - \frac{1}{2}\log\frac{n+\chi}{2\chi} - \frac{3}{2}\chi$.

A problem remains in the $\log 2$ discrepancy in equation (6). This may be dues to the approximate nature of the model or, more likely, to the premature conversion of the problem to a continuous analog.

We have suggested a new problem, how to construct breakpoint graphs reflecting differential rates of inversion and translocation. Our simulation show that the graphs are sensitive to this differential and so analytical work on this problem is important to the eventual utility of our approach in testing the significance of rearrangement inferences.

Acknowledgements

Research supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). DS holds the Canada Research Chair in Mathematical Genomics and is a Fellow of the Evolutionary Biology Program of the Canadian Institute for Advanced Research.

References

1. Eriksen, N. and Hultman, A. 2004. Estimating the expected reversal distance after a fixed number of reversals. Advances of Applied Mathematics 32, 439–453.
2. Kim, J.H. and Wormald, N.C. 2001. Random matchings which induce Hamilton cycles, and Hamiltonian decompositions of random regular graphs, Journal of Combinatorial Theory, Series B 81, 20–44.
3. Mazowita, M., Haque, L. and Sankoff, D. 2006. Stability of rearrangement measures in the comparison of genome sequences. Journal of Computational Biology 13, 554–566.

4. Sankoff, D. 2006. The signal in the genomes. PLoS Computational Biology 2, e35.
5. Sankoff, D. and Haque, L. 2006. The distribution of genomic distance between random genomes. Journal of Computational Biology 13, 1005–1012.
6. Tesler, G. 2002. Efficient algorithms for multichromosomal genome rearrangements. Journal of Computer and System Sciences 65, 587–609.
7. Yancopoulos, S., Attie, O. and Friedberg, R. 2005. Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics 21, 3340 – 3346

Common Intervals and Symmetric Difference in a Model-Free Phylogenomics, with an Application to Streptophyte Evolution

Zaky Adam[1], Monique Turmel[2], Claude Lemieux[2], and David Sankoff[1]

[1] School of Information Technology and Engineering
and Department of Mathematics and Statistics,
University of Ottawa, Ottawa, Canada, K1N 6N5,
{zadam008, sankoff}@uottawa.ca
[2] Département de biochimie et de microbiologie,
Université Laval, Québec, Canada, G1K 7P4,
{monique.turmel, claude.lemieux}@rsvs.ulaval.ca

Abstract. The common intervals of two permutations on n elements are the subsets of terms contiguous in both permutations. They constitute the most basic representation of conserved local order. We use d, the size of the symmetric difference (the complement of the common intervals) of the two subsets of $2^{\{1,\cdots,n\}}$ thus determined by two permutations, as an evolutionary distance between the gene orders represented by the permutations. We consider the Steiner Tree problem in the space $(2^{\{1,\cdots,n\}}, d)$ as the basis for constructing phylogenetic trees, including ancestral gene orders. We extend this to genomes with unequal gene content and to genomes containing gene families. Applied to streptophyte phylogeny, our method does not support the positioning of the complex algae *Charales* as a sister group to the land plants.

1 Introduction

Phylogenetics based on gene order has used two types of objective functions for optimizing the inferred ancestral nodes. One is based on breakpoints, or non-conserved adjacencies in the gene orders of two genomes, dating from 1997 [15] and related to the quantitative pre-genomics literature on conserved segments [12]. The second is based on the number of inversions or other genome rearrangements intervening between two genomes [18,7,19,6].

The scope of a genome rearrangement operation may be unrestricted across the genome; a breakpoint is a very local structure. To emphasize local similarities spanning regions larger than a single breakpoint within a rearrangement analysis, Bergeron and colleagues have integrated more general notions of conserved intervals and common intervals with rearrangements [3,4]. Further, they used these integrated concepts in phylogeny, including the reconstruction of ancestral gene orders [1,2].

In this paper we put even more importance on common intervals, basing a phylogenetic reconstruction method solely on them. This analysis is model-free,

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 63–74, 2006.

in the sense that it dispenses completely with assumptions or considerations, probabilistic or combinatorial, about specific processes involved in rearranging genomes. The objective function we will optimize is simply the sum over the tree branches of the symmetric difference between the two sets of intervals associated with the genomes at the two ends of the branch. The motivation is simply that the more related two genomes are, the more common intervals they will have, while as evolutionary distance increases, more and more intervals from each will be lacking from the other.

Optimizing a tree on the space of subsets of the power set of $\{1, \cdots, n\}$ using dynamic programming [8,9,17] is easy; the set of intervals representing a genome, however, is constrained to be compatible with a permutation. We do not know the complexity of optimizing the ancestral nodes of a tree under this constraint, but conjecture that it is hard. Thus we model our optimization heuristic after the unconstrained case, and add the constraint in the traceback of the algorithm, where a greedy procedure is used to test the addition of possibly conflicting intervals to the subset of intervals representing a genome, represented by a PQ-tree.

Our methods carry over directly to the case where some or all genomes do not contain the full set of genes. All we require is a preliminary assignment of genes to ancestral nodes, rapidly achieved by dynamic programming, and an adjustment of a test for identity of two intervals to be restricted to the reduced intervals containing only the genes in common in the two genomes. The same sort of extension of the model works when duplicate genes and gene families are allowed, but only under certain conditions; the general case will require further work.

We apply our method to the controversial questions of streptophyte phylogeny and recover results comparable to previous work on both gene order phylogeny and DNA sequence-based phylogeny.

2 Notation and Definitions

For a permutation Π on $(1, \cdots, n)$, we write $\Pi = (\pi(1), \cdots, \pi(n))$. Let $\mathcal{S} = \{\Pi_1, \cdots, \Pi_N\}$ be a set of such permutations. For each Π, the interval set determined by $\pi(h)$ and $\pi(k)$, for $1 \leq h < k \leq n$, is $\{\pi(h), \pi(h+1), \cdots, \pi(k)\}$. For each $J = 1, \cdots, N$, let $\mathcal{I}_J \subset 2^{\{1,\cdots,n\}}$ be the $\binom{n}{2}$ interval sets of $2^{\{1,\cdots,n\}}$ determined by Π_J. We define the projection $B(\Pi_J) = \mathcal{I}_J$. The set of common intervals of Π_J and Π_K is $B(\Pi_J) \cap B(\Pi_K) = \mathcal{I}_J \cap \mathcal{I}_K$, the intersection of the $\binom{n}{2}$ interval sets defined by Π_J and those defined by Π_K.

Let $X \subseteq \mathcal{I}_J$ and $Y \subseteq \mathcal{I}_K$. We say X is compatible with $B(\Pi_J)$ and Y is compatible with $B(\Pi_K)$ and define the metric

$$\begin{aligned} d(X, Y) &= |X \cup Y \setminus X \cap Y| \\ &= |X| + |Y| - 2|X \cap Y|. \end{aligned} \tag{1}$$

In particular,

$$\frac{1}{2} d(\mathcal{I}_J, \mathcal{I}_K) = \binom{n}{2} - |\mathcal{I}_J \cap \mathcal{I}_K|. \tag{2}$$

Note that not all subsets of $2^{\{1,\cdots,n\}}$ are permutation-compatible, i.e., are subsets of $B(\Pi)$ for some permutation Π. E.g., $\{\{1,2\},\{2,3\},\{2,4\}\}$ is not permutation-compatible. For any permutation-compatible X, we define $B^{-1}(X)$ to be the set of permutations Π for which $X \subseteq B(\Pi)$.

3 The Steiner Tree Problem for Common Intervals

The Steiner tree problem with input $\mathcal{S}$ is to find a tree graph $T = (V, E)$, where the vertices in V are permutation-compatible subsets of $2^{\{1,\cdots,n\}}$ and $\{B(\Pi)\}_{\Pi\in\mathcal{S}} \subseteq V$ such that the tree length

$$L(T) = \sum_{XY\in E} d(X, Y) \tag{3}$$

is minimal.

In a Steiner tree, we can distinguish two types of vertex in V, terminal vertices, i.e., of degree 1, which are all projections of permutations in $\mathcal{S}$, and non-terminal, or "ancestral", vertices, some or all of which, the "unknown vertices" may be projections of permutations not in $\mathcal{S}$ or other permutation-compatible subsets of $2^{\{1,\cdots,n\}}$. We require that unknown vertices have degree 3 or more, a condition which, by the metric property, does not affect the minimal value of L in (3).

The search for the Steiner tree is often divided into two problems, the inner, or "small", problem, and the outer, or "big", problem. The big problem is essentially a search through the set of all trees satisfying the above description. For each tree examined during this search, the small problem is to optimally identify each of the unknown vertices in V as the projection of some permutation on $(1, \cdots, n)$ or some other permutation-compatible subset of $2^{\{1,\cdots,n\}}$.

4 The General Dynamic Programming Solution for the Small Problem

A general method for the small problem, i.e., for optimizing the position of the ancestral nodes of a tree in a metric space was given in [17].

Condition 1. It suffices to consider trees where there is a path between any two unknown vertices not passing through a vertex in $\mathcal{S}$; othewise the problem decomposes into subproblems an obvious way.

The solution requires choosing, arbitrarily, one of the unknown vertices r as the "root", and directing all edges in E away from the root. We write $v \to u$ for an edge directed from ancestor v to "daughter" u. For any given position of the ancestral vertices, let

$$l(v) = \sum_{v\to u} l(u) + d(u, v), \tag{4}$$

with initial condition $l(v) = 0$ if v is a terminal vertex. Note that the tree structure and Condition 1 ensure that recurrence (4) determines $l(v)$ for all non-terminal vertices. Then for any tree T with specified positions of the ancestral vertices we may rewrite (3) as

$$L(T) = l(r). \tag{5}$$

For a Steiner tree, for a given position of v, we obtain from (4)

$$l(v) = \sum_{v \to u} \min_{u} [l(u) + d(u, v)]. \tag{6}$$

If all the minimizing u are stored at each application of (6), then a Steiner tree T may be recovered by tracing back the recurrence from the root to the terminal nodes.

Depending on the metric space, the minimizing u in (6) may be more or less difficult to calculate. Such is the case that interests us here, where the vertices are projections of permutations, and the search for optimizing u is not straightforward. However, we can easily solve a closely related problem, which provides a lower bound on $L(T)$ and suggests how to solve the problem on permutations.

5 Embedding the Small Problem for Permutations in a Larger Space

For a given $\mathcal{S}$ consider the Steiner tree problem with input $\{B(\Pi_1), \cdots, B(\Pi_N)\}$ in the metric space (M_n, d), where $M_n = 2^{2^{\{1,\cdots,n\}}}$, the set of all subsets of the power set of $\{1, \cdots, n\}$, and d is as before.

The set M_n is larger than $\mathcal{P}_n$, the set of all permutation-compatible subsets of $2^{\{1,\cdots,n\}}$, in that there are elements of $X \in M_n$ which are not of the form $X \subseteq B(\Pi)$ for any permutation Π.

Example 1. Let $X = \{\{1, 2\}, \{1, 3\}, \{1, 4\}\}$. Then there is no permutation Π for which $X \subseteq B(\Pi)$ for any permutation Π.

Then the solution to the Steiner tree problem in (M_n, d) is a lower bound to the solution of the problem in $(\mathcal{P}_n, d)$.

The Steiner tree problem in $(\mathcal{P}_n, d)$ is harder than in (M_n, d) because the intervals sets of a permutation, or sets compatible with a permutation, are highly constrained, whereas in (M_n, d) it suffices to treat each subset of $\{1, \cdots, n\}$ separately. To see this, note that the symmetric difference between two points $X, Y \in M_n$ may be written

$$d(X, Y) = \sum_{\sigma \subset \{1,\cdots,n\}} |\chi_\sigma(X) - \chi_\sigma(Y)|, \tag{7}$$

where χ_σ is the indicator function of σ. Then, writing $X(v)$ for the element of M_n associated with vertex v of a tree, the length of any tree T satisfies

$$L(T) = \sum_{uv \in E} d(X(u), X(v)) \tag{8}$$

$$= \sum_{uv \in E} \sum_{\sigma \subset \{1,\cdots,n\}} |\chi_\sigma(X(u)) - \chi_\sigma(X(v))| \tag{9}$$

$$= \sum_{\sigma \subset \{1,\cdots,n\}} \sum_{uv \in E} |\chi_\sigma(X(u)) - \chi_\sigma(X(v))| \tag{10}$$

so that the minimization of $L(T)$ may be done component-wise, i.e., separately for each $\sigma \subset \{1, \cdots, n\}$. In Section 6 then, we will discuss only whether or not each ancestral vertex contains σ or not, which can be phrased as the tree optimization problem for a single zero-one character.

6 The Small Problem for a Single Zero-One Variable

Consider that the data at each terminal vertex consist of either a zero or a one, representing the presence or absence of a set σ. Dynamic programming for ancestral node optimization requires two passes. In the forward pass, from the terminal nodes towards the root, the value of the variable (the presence or absence of σ) may be established definitely at some ancestral vertices, while at other vertices it is left unresolved until the second, "traceback" pass, when any multiple solutions are also identified.

Suppose ancestral vertex v has p daughter vertices $u_1, \cdots, u_p$, where $p \geq 2$. (If $v \neq r$, then v has has degree $p + 1$, if $v = r$, then v has degree $p \geq 3$.) In practice, it usually suffices to allow only $p = 2$ for ancestral vertices and $p = 3$ for the root (binary trees), and indeed our algorithm is programmed for this case. Nevertheless, in this section we will give the general solution, which may be required in some contexts, and is of mathematical interest in any case. In the next section we will discuss the simplification to binary trees.

Suppose for each daughter u_i, we have already decided whether $\sigma \in u_i$ definitely, possibly, or definitely not. Let

$$q(\sigma) = \#\{i | \sigma \in u_i \text{ definitely}\} \tag{11}$$

$$Q(\sigma) = \#\{i | \sigma \in u_i \text{ definitely or possibly}\}. \tag{12}$$

If $v \neq r$ and $q(\sigma) \geq \frac{p}{2} + 1$, then $\sigma \in v$ definitely. This is true because then

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \leq \frac{p}{2} - 1 \tag{13}$$

whereas if σ were not in v,

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \geq \frac{p}{2} + 1 \tag{14}$$

which is clearly not optimal, no matter whether σ is in the ancestor of v or not.

On the other hand, if $v \neq r$ and $Q(\sigma) \leq \frac{p}{2} - 1$, then definitely $\sigma \notin v$. This is true because then

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \leq \frac{p}{2} - 1 \tag{15}$$

whereas if σ were in v,

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \geq \frac{p}{2} + 1 \tag{16}$$

which is clearly not optimal, no matter whether σ is in the ancestor of v or not.

This leaves the cases where both

$$q(\sigma) < \frac{p}{2} + 1 \tag{17}$$

$$Q(\sigma) > \frac{p}{2} - 1, \tag{18}$$

where we say $\sigma \in v$ possibly.

As for r, if $q(\sigma) > \frac{p}{2}$, then $\sigma \in r$ definitely; if $Q(\sigma) < \frac{p}{2}$, then $\sigma \notin r$ definitely; otherwise $\sigma \in r$ possibly.

While applying the traceback of the recurrence, we reassign the "possible" memberships in ancestral vertices either to definite memberships or to definite exclusions. For ancestral vertex v, whether or not $\sigma \in t$, the ancestor of v, has no bearing if $\sigma \in v$ definitely or $\sigma \notin v$ definitely. Otherwise, if $\sigma \in t$ and $1 + q(\sigma) > \frac{p+1}{2}$, then we reassign $\sigma \in v$ definitely. If $\sigma \notin t$ and $Q(\sigma) < \frac{p+1}{2}$, then we reassign $\sigma \notin v$ definitely.

In the remaining cases if $\sigma \in t$ and $1 + q(\sigma) \leq \frac{p+1}{2} \leq 1 + Q(\sigma)$, or if $\sigma \notin t$ and $Q(\sigma) \geq \frac{p+1}{2} \geq q(\sigma)$, then we can assign or exclude σ from v, without affecting $L(T)$. Note that if $\sigma \in r$ possibly, we are also free to assign it to r or not at the beginning of the traceback.

Note that while the dynamic programming can find a solution in time linear in N, the number of different solutions may be exponential in N. Considering all possible sets σ, the number of solutions is also exponential in n.

7 Binary Trees

The case of binary branching trees, where $p + 1 \equiv 3$ except at the root where $p = 3$, is somewhat simpler in the traceback as there is at most one free choice of assignment, and that is at r. For any other ancestral vertex v, membership or not of σ in t, the ancestor of v, always determines its membership, or not, in v.

Moreover, for the big phylogeny problem, it suffices to search for the optimal binary tree. If the optimal tree is not binary, this will still show up as a binary tree with one or more edges of length $d = 0$.

Nevertheless, even for the small problem, because of the possible choice at r, when we consider all σ, the number of solutions may be exponential in n.

8 Handling Incompatible Subsets in $\mathcal{P}_n$ with PQ-Trees

In the case of binary trees, after the forward pass of the dynamic programming, those σ for which $q(\sigma) = 2$ at any ancestral vertex v must all be in $B(\Pi)$ for some permutation Π, namely any terminal vertex permutation Π for which v is an ancestor. In the case of r, those σ for which $q = 3$ must be in the interval sets of all the terminal vertex permutations in $\mathcal{S}$. In this special case, the solution in (M_n, d) is also a solution for $(\mathcal{P}_n, d)$.

In general, however, this is not the case. In the following example the Steiner tree in M_n is shorter than that in $\mathcal{P}_n$.

Example 2. Let $\mathcal{S} = \{(4,1,2,3),(2,1,3,4),(3,1,4,2)\}$, and suppose T has a single ancestor r.

Then in M_n, we calculate
$q(\{1,2\}) = q(\{1,3\}) = q(\{1,4\}) = 2$,
$q(\{1,2,3\}) = q(\{1,3,4\}) = q(\{1,4,2\}) = 2$,
$q(\{2,3,4\}) = 0$,
$q(\{2,3\}) = q(\{3,4\}) = q(\{4,2\}) = 1$,
so that the only proper subsets in r are $\{1,2\},\{1,3\},\{1,4\},\{1,2,3\},\{1,3,4\}$, $\{1,4,2\}$. Each of the terminal vertices is missing two of these subsets but contains one that is not in r, so that $\sum d = 3 \times (2+1) = 9$.

In $\mathcal{P}_n$, there are multiple solutions to optimizing r, including the permutations $(4,1,2,3),(2,1,3,4)$ or $(3,1,4,2)$ themselves. Each one of the latter shares only two interval sets with each other, so that $\sum d = 12$. Thus the difficulty with working in $\mathcal{P}_n$ is the requirement that the set of subsets at an ancestral vertex has to correspond to a permutation, or at least be compatible with some permutation, while this restriction does not apply in M_n.

There is a data structure particularly well-suited for representing a set of subsets compatible with a permutation, namely the PQ-tree [5]. A PQ-tree is just a rooted tree with terminal vertices $1,\cdots,n$, where the daughter vertices of some special non-terminal vertices may be "ordered", though the left-right or right-left direction of the order is not specified. For an ordered vertex x, the blocks of terms spanned by the various daughters of x must be in the same order for any permutation compatible with the tree, while there is no such constraint on the vertices which are not ordered.

Given a PQ-tree and a new subset X, it is possible to rapidly check whether X is compatible with the other subsets previously used to build the PQ-tree and to update the PQ-tree to include the additional ordering information, if any, in X. PQ-trees have already been utilized in gene order phylogenetics [2,14].

To ensure optimality in our procedure, we would have to interpret u and v in recurrence (6) as PQ-trees. The difficulty would be to carry out the minimization over all possible $u_1,\cdots,u_p$, i.e., to find a constraint limiting the set of possible PQ-trees for u that could be in a solution if a given PQ-tree for v is. This is a problem even for binary trees; the search space of all possible pairs of PQ-trees cannot be reduced to a very limited set as we did in Equation (10) and Section 6. The actual calculation of the symmetric difference induced by two PQ-trees

is not time-consuming. One approach to this problem might be found in the concept of PQR-trees [11,21], which could include a limited number of intervals incompatible with one of the two descendants of v, but we have not explored this.

Instead, we proceed heuristically, constructing a PQ-tree at each vertex only during the traceback. This is motivated by the observation in our test data, that in the traceback it is rare for the following configuration to occur:

- t is the ancestor of v, and v is the ancestor of u_1 and u_2.
- σ_1 and σ_2 are both definitely in the PQ-tree at t, after the traceback step at t.
- σ_i is in u_j only for $i = j$ but not for $i \neq j$, before the traceback.
- either one of σ_1 and σ_2 can be added to PQ-tree defined by the pre-traceback definites at t, but not both.

If this never happens, then we can be sure our result is optimal. If it does happen, we assign as many such σ to the definites of v as possible, using a greedy approach with larger intervals tried before smaller intervals.

9 Unequal Gene Complements

Doing gene order phylogeny on realistic sets of genomes which contain different sets of genes is recognized as a substantially more difficult than the case with identical gene sets in all species [16,20].

There is a natural solution within our framework. There are two steps to this solution.

First, the presence or absence of each gene at each vertex is determined by dynamic programming to minimize the number of gene deletions or insertions across all edges of the tree. This is done gene-by-gene, analogous to the set-by-set procedure in Section 5.

Second, in our main algorithm, when the definites for a vertex v are being decided in the recurrence step, any interval σ in one of its descendants u_1 containing a gene g absent from the other genome u_2 is assigned to be a definite of v if $\sigma \backslash \{g\}$ is present in u_2. This includes the case where $\sigma = \{g, h\}$ and gene h has previously been decided to be present in u_2.

In building the PQ-tree for u_2 during the traceback, if σ is definite for v, and $\sigma \backslash \{g\}$ is a possible for u_2, then it is a candidate for inclusion in the PQ-tree. In calculating the symmetric difference between the sets of intervals from two genomes, we first delete from consideration any gene that is not present in both genomes, and remove any null sets or duplicate sets.

It can be shown that this generalizes the minimization of the sum of symmetric differences across the tree, given the correctness of our dynamic programming solution for gene presence or absence at ancestral vertices. In particular, in the case of identical gene complements, it reduces to our original method.

10 Duplicate Genes

The introduction of gene duplicates and gene families into genome comparisons causes even more difficulty than unequal gene complements. This either calls for changing the analytical framework from comparing permutations to comparing strings, or integrating gene-tree/species-tree methodology into our procedures, or both. The method proposed in this paper, however, formally applies directly to data containing occasional duplicate genes. The construction and comparison of sets of intervals is not complicated by duplicates, except that some convention must be adopted for intervals containing two or more copies of the same gene, i.e., only include one copy per interval or all copies, requiring the same number of copies for identity of two intervals. We did not encounter this problem with our test data, so we have not yet explored it further.

11 Application to Chloroplast Genomes

New sequences of the chloroplast genome allow the exploration of the evolution of the streptophytes, a phylum containing several classes of algae but also the familiar multicellular land plants. Our data includes gene orders from *Mesostigma viride* [10], *Chlorokybus atmophyticus* [unpublished data], *Staurastrum punctulatum* [23], *Zygnema circumcarinatum* [23], *Chaetosphaeridium globosum* [22], *Chara vulgaris* [24], as well as the land plant *Marchantia polymorpha* [13], with 120-140 genes per genome. The first six of these represent five of the six major charophycean (i.e., other than land plants) lineages.

We use the complete gene order of these genomes, including duplicate genes and genes not present in some organisms. There are 148 distinct genes, 35 of which were absent from one or more of the genomes. Five of the genomes had six or eight duplicated genes, largely the same ones in each, while the remaining two genomes had one or zero duplications, respectively.

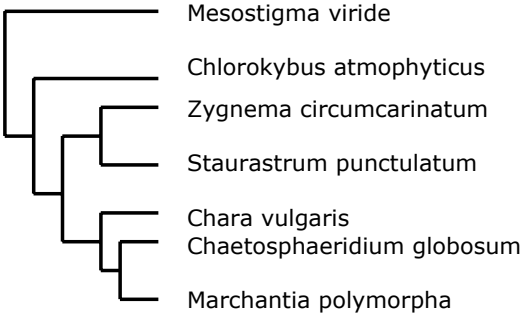


Fig. 1. Best tree for six algae and the land plants. Rooting and branch lengths arbitrary.

There has been some controversy among phycologists about the origin of land plants, in particular whether they represent a sister grouping to the Charales, represented here by *Chara vulgaris* or a sister grouping to the Coleochaetales, represented here by *Chaetosphaeridium globosum*. The best tree we obtained is depicted in Figure 1. We note the correct grouping together of the two Zygnematalean algae *Staurastrum* and *Zygnema*. The tree also correctly depicts the early branching of *Mesostigma* and *Chlorokybus*, though it does not bear on the controversy of whether *Mesostigma* is actually a streptophyte, or whether its origin predates that of the streptophytes in algal evolution. Of particular interest is the grouping of *Marchantia* with *Chaetosphaeridium* instead of with *Chara*. Previous analysis of the same genomes, at both the sequence and gene-order levels, suggested that *Chara* branches early and that the Zygnematale-Coleochaetale lineages group with the land plants [24]. The early branching of the Zygnematales was seen to be a much less parsimonious solution. Nevertheless, the results in Figure 1 represent an additional line of evidence in the still unsettled problem of streptophyte phylogeny.

12 Implementation

Our current implementation of the method includes an exhaustive search of binary trees only for the "big" problem, plus heuristics when N is larger than 9 or 10. The algorithm for the "small" problem is also the binary tree version. We have tested it for values of n in the range of 100-200. The PQ-tree construction notifies when a conflict is detected and resolved, and the pre-calculation of gene content of ancestral nodes is also implemented. The ability to handle duplicate genes is inherent in the algorithm and requires no special consideration. The experimental version of our program will be made available on our lab website.

13 Conclusions

We have proposed and implemented a new method based on common intervals of two or more genomes for constructing a phylogenetic tree. This has the advantage (or disadvantage!) of being independent of any particular model of genome rearrangement or rearrangement weighting. A maximum of emphasis is laid on the commonality of gene order at the local level, the objective function for the tree being merely the sum, over all the tree branches, of the symmetric difference between the two sets of intervals associated with the genomes at the two ends of the branch.

We have only begun to explore the algorithmic possibilities in this approach. The use of PQR-trees during the recurrence part of the algorithm might allow for a global and efficient optimum in the general case. Failing that, more sophisticated heuristics are certainly available for the construction of PQ-trees at ancestral vertices during the traceback.

One issue we have not examined is the interpretation of branch length. The number of intervals at terminal vertex is $O(n^2)$, but in our test data the number

of intervals at ancestral vertices is typically $O(n)$, so that terminal branches would appear unduly distant from the cluster of ancestral vertices, were the untransformed symmetric distance considered meaningful as a clocklike measure of evolution.

Note that this paper deals only with unsigned genomic data. There is no particularly natural way to extend it to signed genomes, except of course to simply drop the sign. Our approach, however, does seem particularly well-suited to situations with gene families, and even to string, instead of permutation, representations of genomes.

Acknowledgements

We would like to thank Laxmi Parida and Annie Chateau for useful discussions on PQ-trees. Research supported in part by a grant to MT,CL and DS from the Natural Sciences and Engineering Research Council of Canada (NSERC) as well as individual grants from this agency. DS holds the Canada Research Chair in Mathematical Genomics and is a Fellow of the Evolutionary Biology Program of the Canadian Institute for Advanced Research.

References

1. Bérard, S., Bergeron, A. and Chauve, C. 2005. Conservation of combinatorial structures in evolution scenarios. Proceedings of the Second RECOMB Satellite Workshop on Comparative Genomics (RECOMB CG 2004), J. Lagergren, ed. Lecture Notes in Computer Science 3388, Springer, 1–14.
2. Bergeron, A., Blanchette, M., Chateau, A. and Chauve, C. 2004. Reconstructing ancestral gene orders using conserved intervals. Proceedings of the First Workshop on Algorithms in Bioinformatics (WABI 2004), Lecture Notes in Computer Science 3240 , Springer, 14–25.
3. Bergeron, A., Heber, S, and Stoye, J. 2002. Common intervals and sorting by reversal: a marriage of necessity. Bioinformatics 18: S54–S63.
4. Bergeron, A. and Stoye, J. 2003. On the similarity of sets of permutations and its applications to genome comparison. Proceedings of the Ninth International Computing and Combinatorics (COCOON 2003), Lecture Notes in Computer Science 2697, Springer, 69–79.
5. Booth K. and Luekar G. 1976. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Journal of Computer and System Sciences 13:335–379.
6. Bourque, G. and Pevzner, P.A. 2002. Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Research 12:26–36.
7. Caprara, A. 2001. On the practical solution of the reversal median problem. Proceedings of the First Workshop on Algorithms in Bioinformatics (WABI 2001), Lecture Notes in Computer Science, Springer, 238–251.
8. Fitch, W. 1971. Toward defining the course of evolution: Minimum change for a specific tree topology. Systematic Zoology 20: 406–416.
9. Hartigan, J.A. 1973. Minimum mutation fits to a given tree. Biometrics 29: 53–65.

10. Lemieux C, Otis C, Turmel M. 2000. Ancestral chloroplast genome in *Mesostigma viride* reveals an early branch of green plant evolution. Nature 403:649–652.
11. Meidanis, J. and Munuera, E.G. 1996. A theory for the consecutive ones property. Proceedings of WSP'97 - Third South American Workshop on String Processing, N. Ziviani, R. Baeza-Yates, eds., Carleton University Press, 194–202.
12. Nadeau, J.H. and Taylor, B. 1984.Lengths of chromosomal segments conserved since divergence of man and mouse. Proc Natl Acad Sci U S A. 81:814–8.
13. Ohyama, K., Fukuzawa, H., Kohchi, T. and 13 co-authors. 1986. Chloroplast gene organization deduced from complete sequence of liverwort Marchantia polymorpha chloroplast DNA. Nature 322:572–574.
14. Parida, L. 2005. A PQ tree-based framework for reconstructing common ancestors under inversions and transpositions. *IBM Research Report* RC 23837.
15. Sankoff, D. and Blanchette, M. 1997. The median problem for breakpoints in comparative genomics. Proceedings of the Third International Computing and Combinatorics Conference (COCOON 1997), T. Jiang and D.T. Lee, eds., Lecture Notes in Computer Science 1276, Springer, 251–263.
16. Sankoff, D., Bryant, D., Deneault, M., Lang, F. and Burger, G. 2000. Early eukaryote evolution based on mitochondrial gene order breakpoints. Journal of Computational Biology 7: 521–535.
17. Sankoff, D. and Rousseau, P. 1975. Locating the vertices of a Steiner tree in an arbitrary metric space. Mathematical Programming 9: 240–246.
18. Sankoff, D., Sundaram, G. and Kececioglu, J. 1996. Steiner points in the space of genome rearrangements. International Journal of the Foundations of Computer Science 7:1–9.
19. Siepel, A. and Moret, B. 2001. Finding an optimal inversion median: experimental results. Proceedings of the First Workshop on Algorithms.in Bioinformatics (WABI 2001), Lecture Notes in Computer Science 2149, Springer, 189–203.
20. Tang, J., and Moret, B.M.E., 2003. Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. Proceedings of the Eighth Workshop on Algorithms and Data Structures (WADS 2003), Lecture Notes in Computer Science 2748, Springer, 37–46.
21. Telles, G.P. and Meidanis, J. 2005. Building PQR trees in almost-linear time. Proceedings of GRACO 2005, P.Feofiloff; C.M.H. de Figueiredo; Y. Wakabayashi, eds., Electronic Notes in Discrete Mathematics 19: 1-416.
22. Turmel M, Otis C, Lemieux C. 2002.The chloroplast and mitochondrial genome sequences of the charophyte *Chaetosphaeridium globosum*: insights into the timing of the events that restructured organelle DNAs within the green algal lineage that led to land plants. Proc Natl Acad Sci U S A. 99:11275–80.
23. Turmel, M., Otis, C. and Lemieux, C. 2005. The complete chloroplast DNA sequences of the charophycean green algae *Staurastrum* and *Zygnema* reveal that the chloroplast genome underwent extensive changes during the evolution of the *Zygnematales*. BMC Biology 3:22.
24. Turmel M, Otis C, Lemieux C.2006. The chloroplast genome sequence of *Chara vulgaris* sheds new light into the closest green algal relatives of land plants. Mol Biol Evol. 23:1324–38.

How Pseudo-boolean Programming Can Help Genome Rearrangement Distance Computation

Sébastien Angibaud[1], Guillaume Fertin[1], Irena Rusu[1], and Stéphane Vialette[2]

[1] Laboratoire d'Informatique de Nantes-Atlantique (LINA), FRE CNRS 2729
Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3 - France
{angibaud, fertin, rusu}@lina.univ-nantes.fr
[2] Laboratoire de Recherche en Informatique (LRI), UMR CNRS 8623
Faculté des Sciences d'Orsay - Université Paris-Sud, 91405 Orsay - France
vialette@lri.fr

Abstract. Computing genomic distances between whole genomes is a fundamental problem in comparative genomics. Recent researches have resulted in different genomic distance definitions: number of breakpoints, number of common intervals, number of conserved intervals, Maximum Adjacency Disruption number (MAD), *etc.* Unfortunately, it turns out that, in presence of duplications, most problems are **NP**–hard, and hence several heuristics have been recently proposed. However, while it is relatively easy to compare heuristics between them, until now very little is known about the absolute accuracy of these heuristics. Therefore, there is a great need for algorithmic approaches that compute exact solutions for these genomic distances. In this paper, we present a novel generic pseudo-boolean approach for computing the exact genomic distance between two whole genomes in presence of duplications, and put strong emphasis on common intervals under the maximum matching model. Of particular importance, we show very strong evidence that the simple LCS heuristic provides very good results on a well-known public benchmark dataset of γ-Proteobacteria.

Keywords: pseudo-boolean programming, genome rearrangement, common intervals, duplication, heuristic.

1 Introduction

Due to the increasing amount of completely sequenced genomes, the comparison of gene order to find conserved gene clusters is becoming a standard approach in comparative genomics. A natural way to compare species is to compare their whole genomes, where comparing two genomes is very often realized by determining a measure of similarity (or dissimilarity) between them.

Several similarity (or dissimilarity) measures between two whole genomes have been recently proposed, such as the number of breakpoints [12,6,2], the number of reversals [6,9], the number of conserved intervals [4], the number of common intervals [5], the Maximum Adjacency Disruption Number (MAD) [13], *etc.* However, in the presence of duplications and for each of the above measures, one has

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 75–86, 2006.

first to disambiguate the data by inferring homologs, *i.e.*, a non-ambiguous mapping between the genes of the two genomes. Up to now, two extremal approaches have been considered : the *exemplar* model and the *maximum matching* model. In the exemplar model [12], for all gene families, all but one occurrence in each genome is deleted. In the maximum matching model [2,8], the goal is to map as many genes as possible. These two models can be considered as the extremal cases of the same generic homolog assignment approach.

Unfortunately, it has been shown that, for each of the above mentioned measures, whatever the considered model (exemplar or matching), the problem becomes **NP**–complete as soon as duplicates are present in genomes [6,2,4,8]; a few inapproximability results are known for some special cases. Therefore, several heuristic methods have been recently devised to obtain (hopefully) good solutions in a reasonable amount of time [3,5]. However, while it is relatively easy to compare heuristics between them, until now very little is known about the absolute accuracy of these heuristics. Therefore, there is a great need for algorithmic approaches that compute exact solutions for these genomic distances.

In the present paper, we introduce a novel generic pseudo-boolean programming approach for computing exact solutions. In this first attempt, we focus on the problem of finding the maximum number of common intervals between two genomes under the maximum matching model. For one, from a computational point of view, this problem (together with MAD) is one of the hardest in our pseudo-boolean framework. For another, this allows us to present with a single example the main idea of our approach: a pseudo-boolean program together with reduction rules. Our approach is in fact more ambitious. Our long term goal is indeed to develop a generic pseudo-boolean approach for the exact computation of various genome distances (number of breakpoints, number of common intervals, number of conserved intervals, MAD, *etc.*) under both the exemplar and the maximum matching models, and use this generic approach on different datasets. The rationale of this approach is threefold:

1. There is a crucial need for new algorithmic solutions providing exact genome distances under both the exemplar and the maximum matching model in order to estimate the accuracy of existing heuristics and to design new efficient biologically relevant heuristics.
2. Very little is known about the relations between the various genome distances that have been defined so far (number of breakpoints, number of common intervals, number of conserved intervals, MAD, *etc.*). We thus propose to extensively compared all these genome distances under both models with a generic pseudo-boolean framework on several datasets.
3. We also plan to further investigate the relations between the exemplar and the maximum matching models. We strongly believe here that, in the light of these comparisons, some biologically relevant *intermediate* model between these two extrema could be defined.

This paper is organized as follows. In Section 2, we present some preliminaries and definitions. We focus in Section 3 on the problem of finding the maximum number of common intervals under the maximum matching model and give a

pseudo-boolean programming approach together with some reduction rules. Section 4 is devoted to experimental results on a dataset of γ-Proteobacteria. Of particular importance, we show strong evidence that the simple LCS heuristic provides very good results on our dataset.

2 Preliminaries

Genomes with duplications are usually represented by signed sequences over the alphabet of *gene families*, where every element in a genome is a *gene*. However, in order to simplify notations, and since common intervals do not depend on the sign given to the genes, we will consider only *unsigned* genomes. Any gene belongs to a gene family, and two genes belong to the same gene family if they have the same label, regardless of the sign. In the sequel, we will be extensively concerned with pairs of genomes. Let G_1 and G_2 be two genomes, and let $a \in \{0, 1\}$. The number of genes in genome G_a is always written n_a. We denote the i-th gene of genome G_a by $G_a[i]$. For any $1 \leq i \leq j \leq n_a$, we write $\mathcal{G}_a(i, j)$ for the set $\{G_a[i], G_a[i+1], \ldots, G_a[j]\}$ and we let $\mathcal{G}_a$ stand for $\mathcal{G}_a(1, n_a)$. In other words, $\mathcal{G}_a(i, j)$ is the set of all distinct genes between positions i and j in genome G_a, while $\mathcal{G}_a$ is the set of all distinct genes in the whole genome G_a. For any gene $\mathbf{g} \in \mathcal{G}_a$ and any $1 \leq i \leq j \leq n_a$, we denote by $\mathsf{occ}_a(\mathbf{g}, i, j)$ the number of occurrences of gene $\mathbf{g}$ in the sequence $(G_a[i], G_a[i+1], \ldots, G_a[j])$. To simplify notations, we abbreviate $\mathsf{occ}_a(\mathbf{g}, 1, n_a)$ to $\mathsf{occ}_a(\mathbf{g})$.

A *matching* $\mathcal{M}$ between genomes G_1 and G_2 is a set of pairwise disjoint pairs $(G_1[i], G_2[j])$, where $G_1[i]$ and $G_2[j]$ belong to the same gene family, *i.e.*, $G_1[i] = G_2[j]$. Genes of G_1 and G_2 that do not belong to any pair of the matching $\mathcal{M}$ are said to be *unmatched* for $\mathcal{M}$. A matching $\mathcal{M}$ between G_1 and G_2 is said to be *maximum* if for any gene family, there are no two genes of this family that are unmatched for $\mathcal{M}$ and belong to G_1 and G_2, respectively. A matching $\mathcal{M}$ between G_1 and G_2 can be seen as a way to describe a putative assignment of orthologous pairs of genes between G_1 and G_2 (see for example [9]).

Let $\mathcal{M}$ be any matching between G_1 and G_2. By first deleting unmatched genes and next renaming genes in G_1 and G_2 according to the matching $\mathcal{M}$, we may now assume that both G_1 and G_2 are duplication free, *i.e.*, G_2 is a permutation of G_1. A *common interval* between G_1 and G_2 is a substring of G_1, *i.e.*, a sequence of consecutive genes of G_1, for which the exact same content can be found in a substring of G_2. It is easily seen that, by first resorting to a renaming procedure, we can always assume that one of the two genomes, say G_1, is the identity permutation, *i.e.*, $G_1 = 1\ 2\ \ldots\ n_1$. For example, let $G = 1\ 2\ 3\ 4\ 5$ and $G_2 = 1\ 5\ 3\ 4\ 2$ then the interval $[3:5]$ of G_1 is a common interval (because $5\ 3\ 4$ occurs as a substring in G_2). Notice that there exist at least $n+1$ ($n = n_1 = n_2$) common intervals between G_1 and G_2 since each individual gene is always a common interval and G_1 itself is also a common interval. This lower bound is tight as shown by $G_1 = 1\ 2\ 3\ 4$ and $G_2 = 2\ 4\ 1\ 3$. Furthermore, if $G_1 = G_2$ the number of common intervals between G_1 and G_2 is $\frac{n(n+1)}{2}$, where $n = n_1 = n_2$, *i.e.*, each possible substring of G_1 is a common interval.

3 An Exact Algorithm for Maximizing the Number of Common Intervals

3.1 Pseudo-boolean Models

A Linear Pseudo-boolean (LPB) program is a linear program [14] where all variables are restricted to take values of either 0 or 1. For one, LPB programs are viewed by the linear programming community as just a domain restriction on general linear programming. For another, from a satisfiability (SAT) point of view, pseudo-boolean constraints can be seen as a *generalization* of clauses providing a significant extension of purely propositional constraints [7,10].

Conventionally, LPB problems are handled by generic Integer Linear Programming (ILP) solvers. The drawback of such an approach is that generic ILP solvers typically ignore the boolean nature of the variables. Alternatively, LPB decision problems could be encoded as SAT instances in pure CNF (Conjunctive Normal Form), *i.e.*, conjunction of disjunctions of boolean literals, which are then solved by any of the highly specialized SAT approaches. However the number of clauses required for expressing the LPB constraints is prohibitively large. Moreover a pure CNF encoding may prevent the solver from pruning the search space effectively [7]. Boolean satisfiability solvers available today are the result of decades of research and are deemed to be among the faster **NP**–complete problem specific solvers. The latest generation of SAT solver generally have three key features (randomization of variable selection, backtracking search and some form of clause learning) and usually run in reasonable time (even for very large instances).

A number of generalizations of SAT solvers to LPB solvers have been proposed (`Pueblo` [15], `Galena` [7], `OPBDP` [1] and more). We decided to use for our tests the `minisat+` LPB solver [10] because of its good results during PB evaluation 2005 (special track of the SAT COMPETITION 2005).

3.2 Common Intervals

We propose in Figure 1 a pseudo-boolean program for computing the maximum number of common intervals between two genomes under the maximum matching model in the presence of duplications (we assume here that each gene $\mathbf{g} \in \mathcal{G}_1 \cup \mathcal{G}_2$ occurs both in G_1 and in G_2).

Program `Common-Intervals-Matching` is clearly a pseudo-boolean program, *i.e.*, a $(0,1)$-linear program. Roughly speaking, the boolean variables are divided in two sets: true setting of variables in C denote possible common intervals between G_1 and G_2, while true setting of variables in X denote the mapping, *i.e.*, matching, between G_1 and G_2. We now turn to describing the constraints. Constraints in (`C.01`) and in (`C.02`) deal with consistency of the mapping: each gene of G_1 is mapped to at most one gene of G_2, and conversely (some genes need indeed to be deleted in case of unbalanced families). Constraints in (`C.03`) ensure that each common interval is counted exactly once. The key idea here is to impose an "*active border*" property, *i.e.*, if variable $c_{k,\ell}^{i,j}$ is set to 1 then genes

Program `Common-Intervals-Matching`

objective:

$$\text{maximize} \sum_{c^{i,j}_{k,\ell} \in A} c^{i,j}_{k,\ell}$$

variables:

$$C = \{c^{i,j}_{k,\ell} : 1 \leq i \leq j \leq n_1 \wedge 1 \leq k \leq \ell \leq n_2\}$$
$$X = \{x^i_k : 1 \leq i \leq n_1 \wedge 1 \leq k \leq n_2 \wedge G_1[i] = G_2[k]\}$$

subject to:

(`C.01`) $\forall i = 1, 2, \ldots, n_1, \quad \sum_{\substack{1 \leq k \leq n_2 \\ G_1[i]=G_2[k]}} x^i_k \leq 1$

(`C.02`) $\forall k = 1, 2, \ldots, n_2, \quad \sum_{\substack{1 \leq i \leq n_1 \\ G_1[i]=G_2[k]}} x^i_k \leq 1$

(`C.03`) $\forall c^{i,j}_{k,\ell} \in C, \quad 4\, c^{i,j}_{k,\ell} - \sum_{\substack{k \leq r \leq \ell \\ G_1[i]=G_2[r]}} x^i_r - \sum_{\substack{k \leq s \leq \ell \\ G_1[j]=G_2[s]}} x^j_s - \sum_{\substack{i \leq p \leq j \\ G_1[p]=G_2[k]}} x^p_k - \sum_{\substack{i \leq q \leq j \\ G_1[q]=G_2[\ell]}} x^q_\ell \leq 0$

(`C.04`) $\forall c^{i,j}_{k,\ell} \in C,\ \forall i < p < j,\ \forall 1 \leq r < k, \quad G_1[p] = G_2[r], \quad c^{i,j}_{k,\ell} + x^p_r \leq 1$

(`C.05`) $\forall c^{i,j}_{k,\ell} \in C,\ \forall i < p < j,\ \forall \ell < r \leq n_2,\ G_1[p] = G_2[r], \quad c^{i,j}_{k,\ell} + x^p_r \leq 1$

(`C.06`) $\forall c^{i,j}_{k,\ell} \in C,\ \forall k < r < \ell,\ \forall 1 \leq p < i, \quad G_1[p] = G_2[r], \quad c^{i,j}_{k,\ell} + x^p_r \leq 1$

(`C.07`) $\forall c^{i,j}_{k,\ell} \in C,\ \forall k < r < \ell,\ \forall j < p \leq n_1,\ G_1[p] = G_2[r], \quad c^{i,j}_{k,\ell} + x^p_r \leq 1$

(`C.08`) $\forall \mathbf{g} \in \mathcal{G}_1 \cup \mathcal{G}_2, \quad \sum_{\substack{1 \leq i \leq n_1 \\ G_1[i]=\mathbf{g}}} \sum_{\substack{1 \leq k \leq n_2 \\ G_2[k]=\mathbf{g}}} x^i_k = \min\{\mathtt{occ}_1(\mathbf{g}), \mathtt{occ}_2(\mathbf{g})\}$

domains:

$$\forall x^i_k \in X, \quad x^i_k \in \{0, 1\}$$
$$\forall c^{i,j}_{k,\ell} \in C, \quad c^{i,j}_{k,\ell} \in \{0, 1\}$$

Fig. 1. Program `Common-Intervals-Matching` for finding the maximum number of common intervals between two genomes under the maximum matching model

$G_1[i]$ and $G_1[j]$ must match some distinct genes between positions k and ℓ in G_2, and genes $G_2[k]$ and $G_2[\ell]$ must match some distinct genes between positions i and j in G_1. Constraints in (`C.04`) to (`C.07`) ensure that if $c^{i,j}_{k,\ell} = 1$ then the interval $[i : j]$ of G_1 and the interval $[k : \ell]$ of G_2 is a common interval according to the mapping induced by the true setting of X. For example, constraints in (`C.04`) ensure that each gene in the interval $[i : j]$ of G_1 is either not mapped or is mapped to a gene in the interval $[k : \ell]$ of G_2 (thanks to constraints in (`C.01`), (`C.02`) and (`C.03`), genes at position i and j in G_1 are actually mapped to distinct genes in G_2 if $i < j$ and $c^{i,j}_{k\ell} = 1$). Finally, constraints in (`C.08`) forces the mapping to be a maximum matching between G_1 and G_2.

Proposition 1. *Program* `Common-Intervals-Matching` *correctly computes the maximum number of common intervals between G_1 and G_2 under the maximum matching model.*

We briefly discuss here space issues of Program `Common-Intervals-Matching`. First, it is easily seen that $\#C = \Theta(n_1^2\, n_2^2)$ and hence that (`C.03`) is composed of $\Theta(n_1^2\, n_2^2)$ constraints. The number of constraints in (`C.04`) to (`C.07`) however does depend on the number of duplications in the two genomes. Second, $\#X = \mathcal{O}(n_1\, n_2)$. Clearly, the size of the set X determines the number of constraints in (`C.01`) and (`C.02`) and of course strongly depends on the number of duplications in G_1 and G_2. Not surprisingly, the set X turns out to be of moderate size in practice. Finally, the number of constraints in (`C.07`) is clearly linear in the size of the two genomes. We shall soon describe (section 3.3) how to speed-up the program by reducing the number of variables and constraints.

We observe that replacing constraints in (`C.08`) by a new set of constraints (`C.08'`) – see below – in Program `Common-Intervals-Matching` results in the pseudo-boolean program `Common-Intervals-Exemplar` that computes the maximum number of common intervals between genomes G_1 and G_2 under the exemplar model.

(`C.08'`) $\forall \mathbf{g} \in \mathcal{G}_1 \cup \mathcal{G}_2, \quad \sum_{\substack{1 \le i \le n_1 \\ G_1[i]=\mathbf{g}}} \; \sum_{\substack{1 \le k \le n_2 \\ G_2[k]=\mathbf{g}}} x_k^i = 1$

Interestingly enough, substituting now the constraints in (`C.08`) by a new set of constraints (`C.08''`) – see below – in Program `Common-Intervals-Matching` results in a pseudo-boolean program that computes the maximum number of common intervals between genomes G_1 and G_2 under the following intermediate model: at least one gene in each gene family is mapped. Observe that this model contains both the exemplar model and the maximum matching model as special cases.

(`C.08''`) $\forall \mathbf{g} \in \mathcal{G}_1 \cup \mathcal{G}_2, \quad \sum_{\substack{1 \le i \le n_1 \\ G_1[i]=\mathbf{g}}} \; \sum_{\substack{1 \le k \le n_2 \\ G_2[k]=\mathbf{g}}} x_k^i \ge 1$

3.3 Speeding-Up the Program

We give in this section four rules for speeding-up Program `Common-Intervals-Matching`. In theory, a very large instance may be easy to solve and a small instance hard. However, very often, small hard instances turn out be artificial, *e.g.*, the pigeonhole problem, and hence, in case of practical instances, the running time of a pseudo-boolean solver is most of the time related to the size of the instances. The main idea here is thus to reduce the number of variables and constraints in the program (for ease of exposition we describe our rules as filters on C). More precisely, we give rules that avoid introducing useless variables $c_{k,\ell}^{i,j}$ in C in such a way that the correctness of Program `Common-Intervals-Matching` is maintained by repeated applications of the rules; two of these filters however do modify the correct maximum number of common intervals between the two genomes and thus ask for subsequent modifications in order to obtain the correct solution.

[**Rule 1**] Delete from C all variables $c_{k,k}^{i,i}$, $1 \leq i \leq n_1$ and $1 \leq k \leq n_2$.

[**Rule 1**] does modify the correct number of common intervals between G_1 and G_2, and hence application of this rule asks for subsequent modifications of the number of common intervals. The key idea of [**Rule 1**] is simply to discard common intervals of size 1 from the program. Indeed, we can compute in a preprocessing step the numbers d_1 and d_2 of genes that need to be deleted in G_1 and G_2 for obtaining a maximum matching between the two genomes. Therefore, we know that the resulting genomes will consist in $L = n_1 - d_1 = n_2 - d_2$ genes, where

$$L = \sum_{\mathbf{g} \in \mathcal{G}} \min\{\mathsf{occ}_1(\mathbf{g}), \mathsf{occ}_2(\mathbf{g})\}.$$

But each of these genes will contribute for 1 to the number of common intervals between G_1 and G_2, for any maximum matching. We thus simply delete all these variables and add L to the number of common intervals between G_1 and G_2 found by Program `Common-Intervals-Matching`.

[**Rule 2**] Delete from C all variables $c_{k,\ell}^{i,j}$ for which any of the following conditions holds true:

1. $(\#\{r : k \leq r \leq \ell \wedge G_1[i] = G_2[r]\} = 0) \vee (\#\{s : k \leq s \leq \ell \wedge G_1[j] = G_2[s]\} = 0)$,
2. $(\#\{r : k \leq r \leq \ell \wedge G_1[i] = G_2[r]\} < 2) \wedge (G_1[i] = G_1[j])$,
3. $(\#\{p : i \leq p \leq j \wedge G_2[k] = G_1[p]\} = 0) \vee (\#\{q : i \leq q \leq j \wedge G_2[\ell] = G_1[q]\} = 0)$,
4. $(\#\{p : i \leq p \leq j \wedge G_2[k] = G_1[p]\} < 2) \wedge (G_2[k] = G_2[\ell])$.

[**Rule 2**] is a quickening for constraints in (`C.03`). Indeed, these constraints ensure that each common interval is counted exactly once by the program by forcing the border of each common interval to be active in the computed solution, *i.e.*, genes $G_1[i]$ and $G_1[j]$ match some distinct genes between positions k and ℓ in G_2, and genes $G_2[k]$ and $G_2[\ell]$ match some distinct genes between positions i and j in G_1. Correctness of [**Rule 2**] thus follows from the fact that Program `Common-Intervals-Matching` will always set a variable $c_{k,\ell}^{i,j}$ to 0 if the border property cannot be satisfied (it is assumed here that $i < j$ and $k < \ell$).

[**Rule 3**] Delete from C all variables $c_{k,\ell}^{i,j}$ for which there exists at least one gene $\mathbf{g} \in \mathcal{G}$ such that $|\mathsf{occ}_1(\mathbf{g}, i, j) - \mathsf{occ}_2(\mathbf{g}, k, \ell)| > |\mathsf{occ}_1(\mathbf{g}) - \mathsf{occ}_2(\mathbf{g})|$.

Roughly speaking, [**Rule 3**] avoids us to kill too many genes in a common interval. Indeed, for one, for any $\mathbf{g} \in \mathcal{G}$, $|\mathsf{occ}_1(\mathbf{g}, i, j) - \mathsf{occ}_2(\mathbf{g}, k, \ell)|$ is clearly the minimum number of occurrences of gene $\mathbf{g}$ that need to be deleted if $c_{k,\ell}^{i,j} = 1$, *i.e.*, $[i : j]$ and $[k : \ell]$ form a common interval between the two genomes. For another, for any $\mathbf{g} \in \mathcal{G}$, $|\mathsf{occ}_1(\mathbf{g}) - \mathsf{occ}_2(\mathbf{g})|$ is the number of occurrences of gene $\mathbf{g}$ that need to be deleted in G_1 and G_2 for finding any maximum matching between the two genomes. Correctness of [**Rule 3**] thus follows from the fact that we can certainly not delete more than $|\mathsf{occ}_1(\mathbf{g}) - \mathsf{occ}_2(\mathbf{g})|$ occurrences of gene $\mathbf{g}$.

[**Rule 4**] Delete from C all variables $c_{k,\ell}^{i,j}$ for which the four following conditions hold true:

1. $\forall \mathbf{g} \in \mathcal{G}_1(i,j), \quad \#\mathsf{occ}_1(\mathbf{g}, 1, i-1) + \#\mathsf{occ}_1(\mathbf{g}, j+1, n_1) = 0,$
2. $\forall \mathbf{g} \in \mathcal{G}_2(k,\ell), \quad \#\mathsf{occ}_2(\mathbf{g}, 1, k-1) + \#\mathsf{occ}_2(\mathbf{g}, \ell+1, n_2) = 0,$
3. $\#\mathsf{occ}_1(G_1[i]) \leq \#\mathsf{occ}_2(G_1[i])$ and $\#\mathsf{occ}_1(G_1[j]) \leq \#\mathsf{occ}_2(G_1[j])$,
4. $\#\mathsf{occ}_2(G_2[k]) \leq \#\mathsf{occ}_1(G_2[k])$ and $\#\mathsf{occ}_2(G_2[\ell]) \leq \#\mathsf{occ}_1(G_2[\ell])$.

We first observe that [**Rule 4**] does modify the correct number of common intervals between G_1 and G_2, and hence application of this rule asks for subsequent modifications of the number of common intervals. The rationale of [**Rule 4**] is that, if the four conditions hold true, then $c_{k,\ell}^{i,j}$ will always be set to 1 by Program `Common-Intervals-Matching`. In other words, for any maximum matching between G_1 and G_2, $[i:j]$ and $[k:\ell]$ will form a common intervals. We thus simply delete from C all these variables $c_{k,\ell}^{i,j}$ and add the number of deleted variables by [**Rule 4**] to the number of common intervals between G_1 and G_2 found by Program `Common-Intervals-Matching`. This rule will prove extremely useful for highly conserved regions with localized duplications.

4 Experimental Results

As mentioned earlier, the generic pseudo-boolean approach we propose in this paper can be useful for estimating the accuracy of an heuristic. In that sense, it is necessary to compute the exact results for different datasets, that could be later used as benchmarks to which confront any given heuristic algorithm.

Computing exact results for different datasets and different (dis)similarity measures is a long task, because the problem is **NP**–hard (see for instance [6,2,8]), which implies that there is no guarantee that a computer (even a very powerful one) will ever provide an exact result ; however, the main interest of the pseudo-boolean approach is that, due to several decades of research intended in speeding-up the computation process, this specific problem can be solved in reasonable time, even for very large instances.

We started the computation of exact results concerning common intervals in the maximum matching model by studying the dataset used in [3]. This dataset is composed of 12 complete genomes from the 13 γ-Proteobacteria originally studied in [11]. The thirteenth genome (*V.cholerae*) has not been considered, since it is composed of two chromosomes, and hence does not fit in the model we considered here for representing genomes. More precisely, this dataset is composed of the genomes of the following species:

- *Buchnera aphidicola APS* (Genbank accession number NC_002528),
- *Escherichia coli K12* (NC_000913),
- *Haemophilus influenzae Rd* (NC_000907),
- *Pasteurella multocida Pm70* (NC_002663),
- *Pseudomonas aeruginosa PA01* (NC_002516),
- *Salmonella typhimurium LT2* (NC_003197),

- *Xanthomonas axonopodis pv. citri 306* (NC_003919),
- *Xanthomonas campestris* (NC_0 03902),
- *Xylella fastidiosa 9a5c* (NC_002488),
- *Yersinia pestis CO_92* (NC_003143),
- *Yersinia pestis KIM5 P12* (NC_004088) and
- *Wigglesworthia glossinidia brevipalpis* (NC_004344).

The computation of a partition of the complete set of genes into gene families, where each family is supposed to represent a group of homologous genes, is taken from [3].

The results we have obtained are given in Table 1. Out of the 66 possible pairwise genome comparisons, 39 results have been obtained so far. Despite the fact that the variant and the measure we study is one of the most time consuming (as mentioned in Section 1), the results look promising, since more than half of the results have been computed until now. Moreover, among those 39 values, only 2 of them took several days to be computed, while the others took no more than a few minutes.

Table 1. Number of common intervals in the maximum matching model: exact results obtained by our pseudo-boolean transformation (39 out of 66)

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xanon	Xcamp	Xfast	Ypest-co92
Ecoli	2882										
Haein	1109	2784									
Paeru	1524		2036								
Pmult	1224	3342	3936	2337							
Salty	2849		2820		3376						
Wglos	1275	2328	1085	1558	1214	2335					
Xanon	1183		1471				1225				
Xcamp	1183		1458				1223				
Xfast	979	1877	1295			1981	994				
Ypest-co92	2585		2694		3298		2318			1949	
Ypest-kim	2141		2500		3092	2093				1891	

In addition to being promising because they were obtained in a reasonable amount of time, these results, though still partial, allow us to go further. Indeed, they will allow us (i) to discuss the heuristic used in [3] (that we will denote *ILCS*), (ii) to discuss an improvement we suggest for *ILCS* (that we will denote *IILCS*), and (iii) to compare the results of *IILCS* to the exact results we have obtained via our pseudo-boolean approach. We first start by describing the heuristic used in [3], that we will call *ILCS* (Iterative Longest Common Substring). This heuristic is greedy, and works as follows:

1. Compute the Longest Common Substring (*i.e.*, the longest contiguous word) S of the two genomes, up to a complete reversal. If there are several candidates, pick one arbitrarily
2. Match all the genes of S accordingly
3. Iterate the process until all possible genes have been matched (*i.e.*, we have obtained a maximum matching)

4. Remove, in each genome, all the genes that have not been matched
5. Compute the number of common intervals that have been obtained in this solution

The simple idea behind this heuristic algorithm is that an LCS (up to complete reversal) of length k contains $\frac{k(k+1)}{2}$ common intervals. Hence, finding such exact copies in both genomes intuitively helps to increase the total number of common intervals. The results obtained by the heuristic *ILCS* in [3] are summarized in Table 2. Since the results are not symmetric (*i.e.*, running $ILCS(G_1, G_2)$ on genomes G_1 and G_2 does not necessarily produce the same number of common intervals than running $ILCS(G_2, G_1)$ on the same genomes taken in the other order), we took, for each genome comparison, the best result.

Table 2. Number of common intervals in the maximum matching model: results obtained by the *ILCS* heuristic from [3]

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xanon	Xcamp	Xfast	Ypest-co92
Ecoli	2605										
Haein	1104	2758									
Paeru	1494	3862	1981								
Pmult	1219	3297	3901	2278							
Salty	2641	65634	2794	3826	3327						
Wglos	1267	2102	1078	1496	1204	2083					
Xanon	1147	2485	1446	3788	1626	2613	1175				
Xcamp	1153	2459	1441	3746	1603	2603	1168	106681			
Xfast	975	1828	1285	2332	1457	1956	963	6797	6647		
Ypest-co92	2414	13818	2668	3887	3237	15007	2037	2395	2358	1810	
Ypest-kim	1943	13762	2460	3757	3027	14770	1846	2471	2446	1854	242963

A deeper study of the *ILCS* heuristic led us to suggest an improvement to it, in the form of a new heuristic, that we call *IILCS* (Improved Iterative Longest Common Substring). The only difference between *IILCS* and *ILCS* is that, before searching for a Longest Common Substring (up to a complete reversal), we "*tidy*" the two genomes, in the sense that we remove, in each genome and at each iteration, the genes for which we know they will not be matched in the final solution (this is simply done by counting, at each iteration, the number of unmatched genes of each gene family). This actually allows to possibly find longer Longest Common Substrings at each iteration, and always gives better results on the studied dataset (except in one case where the result is the same for both heuristics, see Table 3). On average, over the 66 pairwise genome comparisons, *IILCS* improves by 2.6% the number of common intervals that are found. The results for *IILCS* are summarized in Table 3. Similarly to *ILCS*, the results are not symmetric, thus we took, for each genome comparison, the best result.

The most interesting and surprising result, which we were able to point thanks to our pseudo-boolean transformation and the exact results obtained from it, is that heuristic *IILCS* appears to be *very* good on the dataset we studied. Indeed, out of the 39 instances for which we have computed the exact results, *IILCS* returns the *optimal result* in 7 cases, and returns a number of common intervals

Table 3. Number of common intervals in the maximum matching model: results obtained by the *IILCS* heuristic

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xanon	Xcamp	Xfast	Ypest-co92
Ecoli	2869										
Haein	1109	2775									
Paeru	1518	3976	2018								
Pmult	1224	3329	3887	2321							
Salty	2849	66025	2809	3956	3350						
Wglos	1267	2186	1085	1508	1211	2274					
Xanon	1183	2540	1468	3952	1644	2685	1198				
Xcamp	1183	2524	1455	3898	1621	2675	1196	108347			
Xfast	979	1849	1293	2365	1464	1974	973	6890	6732		
Ypest-co92	2541	14364	2686	3989	3268	15192	2307	2482	2433	1816	
Ypest-kim	2124	14126	2487	3859	3037	15451	2091	2557	2509	1863	261345

that is more than 99% of the optimal number for 18 other cases. The "worse" result that *IILCS* provides is 93.17% away from the optimal (Ypest-co92/Xfast). On average, over the 39 pairwise comparisons for which we have exact results, *IILCS* performs very well, since it gives a number of common intervals that is 98.91% of the optimal number.

This result comes as a surprise, because, despite being extremely simple and fast, *IILCS* appears to be very good on this dataset. Hence, this strongly suggests that computing common intervals in the maximum matching model can simply be undertaken using *IILCS* while remaining accurate, thus validating this heuristic.

5 Conclusion

In this paper, we have introduced a novel and original method that helps speeding-up computations of exact results for comparing whole genomes containing duplicates. This method makes use of pseudo-boolean programming. Our approach is very general, and can handle several (dis)similarity measures (breakpoints, common intervals, conserved intervals, MAD, *etc.*) under several possible models (exemplar model, maximum matching model, but also most variants within those two extrema). An example of such an approach (common intervals under the maximum matching model) has been developed, in order to illustrate the main ideas of the pseudo-boolean transformation framework that we suggest. Experiments have also been undertaken on a dataset of γ-Proteobacteria, showing the validity of our approach, since already 39 results (out of 66) have been obtained in a limited amount of time. Moreover, those preliminary results have allowed us to state that the new *IILCS* heuristic provides excellent results on this dataset, hence showing its validity and robustness. On the whole, those preliminary results are very encouraging.

There is still a great amount of work to be done, and some of it is being undertaken by the authors at the moment. Among other things, one can cite:

– Implementing and testing all the possible above mentioned variants, for all the possible above mentioned (dis)similarity measures,

- For each case, determine strong and relevant rules for speeding-up the process by avoiding the generation of too many clauses and variables (a pre-processing step that should not be underestimated),
- Obtaining exact results for each of those variants and measures, for different datasets, that could be later used as benchmarks for validating (or not) possible heuristics, but also the measures themselves, or even the models.

References

1. P. Barth. A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization. Technical Report MPI-I-95-2-003, Max Planck Institut Informatik, 2005. 13 pages.
2. G. Blin, C. Chauve, and G. Fertin. The breakpoint distance for signed sequences. In *Proc. 1st Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets)*, pages 3–16. KCL publications, 2004.
3. G. Blin, C. Chauve, and G. Fertin. Genes order and phylogenetic reconstruction: Application to γ-proteobacteria. In *Proc. 3rd RECOMB Comparative Genomics Satellite Workshop*, volume 3678 of *LNBI*, pages 11–20, 2005.
4. G. Blin and R. Rizzi. Conserved intervals distance computation between non-trivial genomes. In *Proc. 11th Annual Int. Conference on Computing and Combinatorics (COCOON)*, volume 3595 of *LNCS*, pages 22–31, 2005.
5. G. Bourque, Y. Yacef, and N. El-Mabrouk. Maximizing synteny blocks to identify ancestral homologs. In *Proc. 3rd RECOMB Comparative Genomics Satellite Workshop*, volume 3678 of *LNBI*, pages 21–35, 2005.
6. D. Bryant. The complexity of calculating exemplar distances. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 207–212. 2000.
7. D. Chai and A. Kuehlmann. A fast pseudo-boolean constraint solver. pages 830–835, 2003.
8. C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Genomes containing duplicates are hard to compare. In *Proc Int. Workshop on Bioinformatics Research and Applications (IWBRA)*, volume 3992 of *LNCS*, pages 783–790, 2006.
9. X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–315, 2005.
10. N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.
11. E. Lerat, V. Daubin, and N.A. Moran. From gene tree to organismal phylogeny in prokaryotes: the case of γ-proteobacteria. *PLoS Biology*, 1(1):101–109, 2003.
12. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.
13. D. Sankoff and L. Haque. Power boosts for cluster tests. In *Proc. 3rd RECOMB Comparative Genomics Satellite Workshop*, volume 3678 of *LNBI*, pages 11–20, 2005.
14. A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, 1998.
15. H.M. Sheini and K.A. Sakallah. Pueblo: A hybrid pseudo-boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:165–189, 2006.

Sorting by Translocations Via Reversals Theory

Michal Ozery-Flato and Ron Shamir

School of Computer Science, Tel-Aviv University, Tel Aviv 69978, Israel
{ozery, rshamir}@post.tau.ac.il

Abstract. The understanding of genome rearrangements is an important endeavor in comparative genomics. A major computational problem in this field is finding a shortest sequence of genome rearrangements that "sorts" one genome into another. In this paper we focus on sorting a multi-chromosomal genome by translocations. We reveal new relationships between this problem and the well studied problem of sorting by reversals. Based on these relationships, we develop two new algorithms for sorting by translocations, which mimic known algorithms for sorting by reversals: a score-based method building on Bergeron's algorithm, and a recursive procedure similar to the Berman-Hannenhalli method. Though their proofs are more involved, our procedures for translocations match the complexities of the original ones for reversals.

1 Introduction

For over a decade, much effort has been put into large-scale genome sequencing projects. Analysis of biological sequence data that have accumulated so far showed that genome rearrangements play an important role in the evolution of species. A major computational problem in the research of genome rearrangements is finding a most parsimonious sequence of rearrangements that transforms one genome into the other. This is called the *genomic sorting problem*, and the corresponding number of rearrangements is called the *genomic distance* between the two genomes. Genomic sorting gives rise to a spectrum of fascinating combinatorial problems, each defined by the set of allowed rearrangement operations and by the representation of the genomes.

In this paper we focus on the problem of sorting by translocations. We reveal new similarities between sorting by translocations and the well studied problem of sorting by reversals. The study of the problem of sorting by translocations is essential for the full comprehension of any genomic sorting problem that permits translocations. Below we review the relevant previous results and summarize our results. Formal definitions are provided on the next section.

Following the pioneering work by Nadeau and Taylor [11], reversals and translocations are believed to be very common in the evolution of mammalian species. *Reversals* (or inversions) reverse the order and the direction of transcription of the genes in a segment inside a chromosome. *Translocations* exchange tails between two chromosomes. A translocation is *reciprocal* if none of the exchanged tails is empty. The genomic sorting problem restricted to reversals (respectively, reciprocal translocations) is referred to as SBR (respectively, SRT).

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 87–98, 2006.

SBR and SRT were both proven to be polynomial. Hannenhalli and Pevzner [7] gave the first polynomial algorithm for SBR and since then other more efficient algorithms and simplifications for the analysis have been presented. Berman and Hannenhalli [4] presented a recursive algorithm for SBR. Kaplan, Shamir and Tarjan [8] simplified the analysis and gave an $O(n^2)$ algorithm for SBR. Using a linear time algorithm by Bader, Moret and Yan [1] for computing the reversal distance, the algorithm of Berman and Hannenhalli can be implemented in $O(n^2)$. A score-based algorithm for SBR was presented by Bergeron [2]. Tannier, Bergeron and Sagot [13] presented an elegant algorithm for SBR that can be implemented in $O(n^{3/2}\sqrt{log(n)})$ using a clever data structure by Kaplan and Verbin [9].

SRT was first introduced by Kececioglu and Ravi [10] and was given a polynomial time algorithm by Hannenhalli [5]. Bergeron, Mixtacki and Stoye [3] pointed to an error in Hannenhalli's proof of the translocation distance formula and consequently in Hannenhalli's algorithm. They presented a new proof followed by an $O(n^3)$ algorithm for SRT. In a recent study [12] we proved that the algorithm of Tannier et al.[13] for SBR can be adapted to solve SRT while preserving the original time complexity (that is $O(n^{3/2}\sqrt{log(n)})$).

It is well known that a translocation on a multi-chromosomal genome can be simulated by a reversal on a concatenation of the chromosomes [6]. However, different translocations require different concatenations. In addition, intra-chromosomal reversals on a concatenation of the chromosomes do not have matching translocations. Thus, from a first glance the similarity between SRT and SBT is rather limited. In [12] we presented the "overlap graph with chromosomes" of two multi-chromosomal genomes, which is an extension of the "overlap graph" of two uni-chromosomal genomes. This auxiliary graph established a new framework for the analysis of SRT that enabled us to adapt the currently fastest algorithm for SBR to SRT [13,12]. In this paper we reveal new relationships between SRT and SBR. Based on these relationships we develop two new algorithms for SRT, which mimic known algorithms for SBR: a score-based method building on Bergeron's algorithm [2], and a recursive procedure similar to the Berman-Hannenhalli method [4]. Though the proofs of the algorithms are more involved than those of their counterparts for SBR, our procedures for translocations match the complexities of the original ones for reversals: the score-based algorithms performs $O(n^2)$ operations on $O(n)$ bit vectors; the recursive algorithm runs in $O(n^2)$ time.

The paper is organized as follows. Section 2 gives the necessary preliminaries. Section 3 presents the score-based algorithm and Section 4 presents the recursive algorithm.

2 Preliminaries

This section provides a basic background for the analysis of SRT. It follows to a large extent the nomenclature and notation of [5,8]. In the model we consider, a *genome* is a set of chromosomes. A *chromosome* is a sequence of genes. A *gene* is identified by a positive integer. All genes in the genome are distinct. When it

appears in a genome, a gene is assigned a sign of plus or minus. For example, the following genome consists of 8 genes in two chromosomes:

$$A_1 = \{(1, -3, -2, 4, -7, 8), (6, 5)\}.$$

The *reverse* of a sequence of genes $I = (x_1, \ldots, x_l)$ is $-I = (-x_l, \ldots, -x_1)$. A *reversal* reverses a segment of genes inside a chromosome. Two chromosomes, X and Y, are *identical* if either $X = Y$ or $X = -Y$. Therefore, *flipping* chromosome X into $-X$ does not affect the chromosome it represents.

A *signed permutation* $\pi = (\pi_1, \ldots, \pi_n)$ is a permutation on the integers $\{1, \ldots, n\}$, where a sign of plus or minus is assigned to each number. If A is a genome with the set of genes $\{1, \ldots, n\}$ then any concatenation π_A of the chromosomes of A is a signed permutation of size n. In the following, we assume without loss of generality that there is a concatenation of the chromosomes in B, π_B, which is identical to the identity permutation. For example,

$$B = \{(1, 2, \ldots, 5), (6, 7, 8)\}.$$

Let $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ be two chromosomes, where X_1, X_2, Y_1, Y_2 are sequences of genes. A *translocation* cuts X into X_1 and X_2 and Y into Y_1 and Y_2 and exchanges segments between the chromosomes. It is called *reciprocal* if X_1,X_2, Y_1 and Y_2 are all non-empty. There are two ways to perform a translocation on X and Y. A *prefix-suffix* translocation switches X_1 with Y_2 resulting in:

$$(\underline{X_1}, X_2), (Y_1, \underline{Y_2}) \Rightarrow (-Y_2, X_2), (Y_1, -X_1).$$

A *prefix-prefix* translocation switches X_1 with Y_1 resulting in:

$$(\underline{X_1}, X_2), (\underline{Y_1}, Y_2) \Rightarrow (Y_1, X_2), (X_1, Y_2).$$

Note that we can mimic a prefix-prefix (respectively, prefix-suffix) translocation by a flip of one of the chromosomes followed by a prefix-suffix (respectively, prefix-prefix) translocation. As was demonstrated by Hannenhalli and Pevzner [6], a translocation on A can be simulated by a reversal on π_A in the following way:

$$(\ldots, X_1, \underline{X_2, \ldots, Y_1}, Y_2, \ldots) \Rightarrow (\ldots, X_1, \underline{-Y_1, \ldots, -X_2}, Y_2, \ldots).$$

The type of translocation depends on the relative orientation of X and Y in π_A (and not on their order): if the orientation is the same, then the translocation is prefix-suffix, otherwise it is prefix-prefix. The segment between X_2 and Y_1 may contain additional chromosomes that are flipped and thus unaffected.

For a chromosome $X = (x_1, \ldots, x_k)$ define $Tails(X) = \{x_1, -x_k\}$. Note that flipping X does not change $Tails(X)$. For a genome A_1 define $Tails(A_1) = \bigcup_{X \in A_1} Tails(X)$. For example:

$$Tails(\{(1, -3, -2, 4, -7, 8), (6, 5)\}) = \{1, -8, 6, -5\}.$$

Two genomes A_1 and A_2 are *co-tailed* if $Tails(A_1) = Tails(A_2)$. In particular, two co-tailed genomes have the same number of chromosomes. Note that if A_2 was obtained from A_1 by performing a reciprocal translocation then $Tails(A_2) = Tails(A_1)$. Therefore, SRT is defined only for genomes that are co-tailed. For the

rest of this paper the word "translocation" refers to a reciprocal translocation and we assume that the given genomes, A and B, are co-tailed.

2.1 The Cycle Graph

Let N be the number of chromosomes in A (equivalently, B). We shall always assume that both A and B contain genes $\{1, \ldots, n\}$. The *cycle graph* of A and B, denoted $G(A, B)$, is defined as follows. The set of vertices is $\bigcup_{i=1}^{n} \{i^0, i^1\}$. For every pair of adjacent genes in B, i and $i+1$, add a grey edge $(i, i+1) \equiv (i^1, (i+1)^0)$. For every pair of adjacent genes in A, i and j, add a black edge $(i, j) \equiv (out(i), in(j))$, where $out(i) = i^1$ if i has a positive sign in A and otherwise $out(i) = i^0$, and $in(j) = j^0$ if j has a positive sign in A and otherwise $in(j) = j^1$. An example is given in Fig. 1. There are $n - N$ black edges and $n - N$ grey edges in $G(A, B)$. A grey edge $(i, i+1)$ is *external* if the genes i and $i+1$ belong to different chromosomes of A, otherwise it is *internal*.

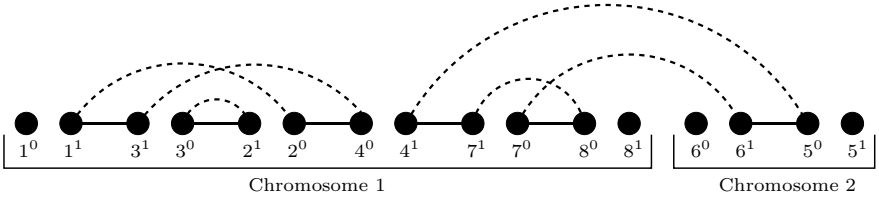


Fig. 1. The cycle graph $G(A_1, B_1)$, where $A_1 = \{(1, -3, -2, 4, -7, 8), (6, 5)\}$ and $B_1 = \{(1, \ldots, 5), (6, 7, 8)\}$. Dotted lines corresponds to grey edges.

Every vertex in $G(A, B)$ has degree 2 or 0, where vertices of degree 0 (isolated vertices) belong to $Tails(A)$ (equivalently, $Tails(B)$). Therefore, $G(A, B)$ is uniquely decomposed into cycles with alternating grey and black edges. Note that the cycle graph is uniquely decomposed into cycles iff A and B are co-tailed. An *adjacency* is a cycle with two edges.

2.2 The Overlap Graph with Chromosomes

Place the vertices of $G(A, B)$ along a straight line according to their order in π_A. Now, every grey edge can be associated with an interval of vertices of $G(A, B)$. Two intervals *overlap* if their intersection is not empty but none contains the other. The *overlap graph with chromosomes* of A and B w.r.t. π_A, denoted $\mathrm{OVCH}(A, B, \pi_A)$, is defined as follows. There are two types of vertices. The first type corresponds to grey edges in $G(A, B)\}$. The second type corresponds to chromosomes of A. Two vertices are connected if their associated intervals of vertices overlap. For example see Fig. 2.

In order to avoid confusion, we will refer to nodes that correspond to chromosomes as "chromosomes" and reserve the word "vertex" for the nodes that correspond to grey edges of $G(A, B)$. A vertex in $\mathrm{OVCH}(A, B, \pi_A)$ is *external* iff there is an edge connecting it to a chromosome, otherwise it is *internal*. Note

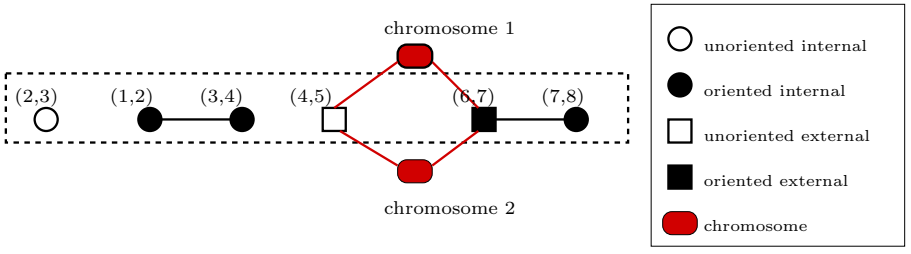


Fig. 2. The overlap graph with chromosomes $\mathrm{OVCH}(A_1, B_1, \pi_{A_1})$, where A_1, B_1 and π_{A_1} are as A_1 and B_1 are the genomes from Fig. 1 and $\pi_{A_1} = (1, -3, -2, 4, -7, 8, 6, 5)$. The graph induced by the vertices within the dashed rectangle is $\mathrm{OV}(A_1, B_1, \pi_{A_1})$.

that the internal/external state of a vertex in $\mathrm{OVCH}(A, B, \pi_A)$ does not depend on π_A (the partition of the chromosomes is known from A). A vertex in the overlap graph is *oriented* if its corresponding edge connects two genes with different signs in π_A, otherwise it is *unoriented*.

Let $\mathrm{OV}(A, B, \pi_A)$ be the subgraph of $\mathrm{OVCH}(A, B, \pi_A)$ induced by the set of vertices that correspond to grey edges (i.e. excluding the chromosomes' vertices). We shall use the word "component" for a connected component of $\mathrm{OV}(A, B, \pi_A)$. The set of components in $\mathrm{OVCH}(A, B, \pi_A)$ can be computed in linear time using an algorithm by Bader, Moret and Yan [1]. A component in $\mathrm{OVCH}(A, B, \pi_A)$ is *external* if at least one of the vertices in it is external, otherwise it is *internal*. A component is *trivial* if it is composed of one internal vertex. A trivial component corresponds to an adjacency. It is not hard to see that the set of internal components in $\mathrm{OVCH}(A, B, \pi_A)$ is independent of π_A. Denote by $\mathcal{IN}(A, B)$ the set of non-trivial internal components in $\mathrm{OVCH}(A, B, \pi_A)$.

2.3 The Reciprocal Translocation Distance

Let $c(A, B)$ denote the number of cycles in $G(A, B)$.

Theorem 1. *[3,5] The reciprocal translocation distance between A and B is $d(A, B) = n - N - c(A, B) + F(A, B)$, where $F(A, B) \geq 0$ and $F(A, B) = 0$ iff $\mathcal{IN}(A, B) = \emptyset$.*

Let Δc denote the change in the number of cycles after performing a translocation on A. Then $\Delta c \in \{-1, 0, 1\}$ [5]. A translocation is *proper* if $\Delta c = 1$. A translocation is *safe* if it does not create any new non-trivial internal component. A translocation ρ is *valid* if $d(A \cdot \rho, B) = d(A, B) - 1$. It follows from Theorem 1 that if $\mathcal{IN}(A, B) = \emptyset$, then every safe proper translocation is necessarily valid.

In a previous paper [12] we presented a generic algorithm for SRT that uses a sub-procedure for solving SRT when $\mathcal{IN}(A, B) = \emptyset$. The generic algorithm focuses on the efficient elimination of the non-trivial internal components. We showed that the work performed by this generic algorithm, not including the sub-procedure calls, can be implemented in linear time. This led to the following theorem:

Theorem 2. *[12] SRT is linearly reducible to SRT with* $IN(A, B) = \emptyset$.

By the theorem above, it suffices to solve SRT assuming that $\mathcal{IN}(A, B) = \emptyset$. Both algorithms that we describe below will make this assumption.

2.4 The Effect of a Translocation on the Overlap Graph with Chromosomes

Let $H = \text{OVCH}(A, B, \pi_A)$ and let v be any vertex in H. Denote by $N(v) \equiv N(v, H)$ the set of vertices that are neighbors of v in H, including v itself (but not including chromosome neighbors). Denote by $\text{CH}(v) \equiv \text{CH}(v, H)$ the set of chromosomes that are neighbors of v in H. Hence if v is external then $|\text{CH}(v)| = 2$, otherwise $\text{CH}(v) = \emptyset$.

Every external grey edge e defines one proper translocation that cuts the black edges incident to e. (Out of the two possibilities of prefix-prefix or prefix-suffix translocations, exactly one would be proper.) For an external vertex v denote by $\rho(v)$ the proper translocation that the corresponding grey edge defines on A. Let $H \cdot \rho(v) = \text{OVCH}(A \cdot \rho(v), B, \pi_A)$. Given two sets S_1 and S_2 define $S_1 \bigoplus S_2 = (S_1 \bigcup S_2) \setminus (S_1 \bigcap S_2)$.

Lemma 1. *[12] Let v be an oriented external vertex in H. Then $H \cdot \rho(v)$ is obtained from H by the following operations. (i) Complement the subgraph induced by $N(v)$ and flip the orientation of every vertex in $N(v)$. (ii) For every vertex $u \in N(v)$ such that the endpoints of u and v share at least one common chromosome, complement the edges between u and $\text{CH}(u) \bigcup \text{CH}(v)$ (In other words $\text{CH}(u, H \cdot \rho(v)) = \text{CH}(u, H) \bigoplus \text{CH}(v, H)$).*

Two overlap graphs with chromosomes are *equivalent* if one can be obtained from the other by a sequence of chromosome flips. For a chromosome X let $\rho(X)$ denote a flip of chromosome X in π_A. Let $H \cdot \rho(X) = \text{OVCH}(A, B, \pi_A \cdot \rho(X))$.

Lemma 2. *[12] $H \cdot \rho(X)$ is obtained from H by complementing the subgraph induced by the set $\{u : X \in \text{CH}(u)\}$ and flipping the orientation of every vertex in it.*

It follows that an unoriented external vertex v in H becomes an oriented (external) vertex in $H \cdot \rho(X)$, where $X \in \text{CH}(v)$.

3 A Score-Based Algorithm

In this section we present a score-based algorithm for SRT when $\mathcal{IN}(A, B) = \emptyset$. This algorithm is similar to an algorithm by Bergeron for SBR [2].

Denote by $N_{\text{IN}}(v)$ and $N_{\text{EXT}}(v)$ the neighbors of v that are respectively internal and external. It follows that $N_{\text{IN}}(v) \bigcup N_{\text{EXT}}(v) \bigcup \{v\} = N(v)$.

Lemma 3. *Let v be an oriented external vertex in H and let w be a neighbor of v. w has no external neighbors in $H \cdot \rho(v)$ iff $N_{\text{EXT}}(w) \subseteq N_{\text{EXT}}(v)$ and $N_{\text{IN}}(v) \subseteq N_{\text{IN}}(w)$.*

For each vertex v in H we define the *score* $|N_{\mathrm{EXT}}(v)|-|N_{\mathrm{IN}}(v)|$. Define $\Delta\mathrm{IN}(H,v)$ as the set of vertices that belong to external components in H but are in non-trivial internal components in $H \cdot \rho(v)$.

Lemma 4. *Let O be a set of oriented external vertices. Let $v \in O$ be a vertex with maximal score in O. Then $O \bigcap \Delta\mathrm{IN}(H,v) = \emptyset$.*

Proof. Assume that $u \in O \bigcap \Delta\mathrm{IN}(H,v)$. Then $u \in N(v,H)$ and by Lemma 3 $N_{\mathrm{EXT}}(u) \subseteq N_{\mathrm{EXT}}(v)$ and $N_{\mathrm{IN}}(v) \subseteq N_{\mathrm{IN}}(u)$. However, since v has the maximal score in O, we get $N_{\mathrm{EXT}}(u) = N_{\mathrm{EXT}}(v)$ and $N_{\mathrm{IN}}(v) = N_{\mathrm{IN}}(u)$. Therefore, u is an isolated internal vertex in $H \cdot \rho(v)$, a contradiction for $u \in \Delta\mathrm{IN}(H,v)$. □

Theorem 3. *Let O be a non-empty set of* all *the oriented external vertices in H that overlap the same pair of chromosomes (i.e. $\mathrm{CH}(u) = \mathrm{CH}(v)$ for every $u,v \in O$). Let $v \in O$ be a vertex that has the maximal score in O. Let S be the set of all the vertices w that satisfy the following conditions in H:*

1. *w is a neighbor of v,*
2. *w is an unoriented external vertex and $\mathrm{CH}(w) = \mathrm{CH}(v)$,*
3. *$N_{\mathrm{EXT}}(w) \subseteq N_{\mathrm{EXT}}(v)$,*
4. *$N_{\mathrm{IN}}(v) \subseteq N_{\mathrm{IN}}(w)$, and*
5. *$O \bigcap N_{\mathrm{EXT}}(v) \subseteq N_{\mathrm{EXT}}(w)$.*

If $S = \emptyset$ then $\rho(v)$ is safe. Otherwise, let $w \in S$ be a vertex that has a maximal score in $H \cdot \rho(X)$, where $X \in \mathrm{CH}(v)$. Then $\rho(w)$ is safe.

Proof. Suppose $S = \emptyset$ and assume that v is not safe. Let $w \in \Delta\mathrm{IN}(H,v)$ be a neighbor of v in H. It follows from Lemma 1 that $\mathrm{CH}(w) = \mathrm{CH}(v)$. It follows from Lemmas 3 and 4 that $w \in S$, a contradiction.

Suppose $S \neq \emptyset$. Let $O_1 = O \bigcap N_{\mathrm{EXT}}(v)$. Then there are all possible edges between S and O_1 in H (last condition). Let $H' = H \cdot \rho(X)$, where $X \in \mathrm{CH}(v)$. In H' all the vertices in S are oriented. Moreover, there are no edges between S and $O_1 \bigcup \{v\}$ in H'. It follows that $O_1 \bigcup \{v\}$ remains external after performing a translocation on any vertex in S. Let $w \in S$ be a vertex with maximal score in S and assume $\Delta\mathrm{IN}(H',w) \neq \emptyset$. Let $u \in \Delta\mathrm{IN}(H',w)$ be a neighbor of w in H'. Then u satisfies: *(i)* $\mathrm{CH}(u) = \mathrm{CH}(w)$ and *(ii)* there are no edges between u and $O_1 \bigcup \{v\}$ in H'. Moreover, u is oriented in H' since otherwise $u \in O_1$ and thus could not be a neighbor of w. It follows that u satisfies conditions 1, 2 and 5 in H. However, by Lemma 4 it follows that $u \notin S$. Hence there are two possible cases:

Case 1: $N_{\mathrm{EXT}}(u) \nsubseteq N_{\mathrm{EXT}}(v)$ in H (i.e. condition 3 is not satisfied). Suppose $z \in N_{\mathrm{EXT}}(u)$ and $z \notin N_{\mathrm{EXT}}(v)$ in H. Then $z \notin N_{\mathrm{EXT}}(w)$ in H (condition 3).

Case 1.*a*: $X \notin \mathrm{CH}(z)$. Then in H': $z \in N_{\mathrm{EXT}}(u)$ and $z \notin N_{\mathrm{EXT}}(w)$. Then according to Lemma 3, z has an external neighbor in $H' \cdot \rho(w)$, a contradiction.

Case 1.*b*: $X \in \mathrm{CH}(z)$. Then in H': $z \notin N(u)$, $z \in N(v)$, $z \in N(w)$. Therefore, in $H' \cdot \rho(w)$ the path u,z,v exists, a contradiction (since v is external in $H' \cdot \rho(w)$).

Case 2: $N_{\mathrm{IN}}(v) \nsubseteq N_{\mathrm{IN}}(u)$ in H (i.e. condition 4 is not satisfied). Then there exists $x \in N_{\mathrm{IN}}(v)$, $x \notin N_{\mathrm{IN}}(u)$ in H'. It follows from condition 4 that $x \in N_{\mathrm{IN}}(w)$

in H'. Since x is internal, all its edges exist in H'. It follows from Lemma 3 that u has an external neighbor (x) in $H' \cdot \rho(w)$, a contradiction. □

Theorem 3 immediately implies an $O(n^3)$ algorithm that can be implemented using operations on bit vectors, in a similar manner to the implementation of the algorithm of Bergeron [2] for SBR. The algorithm is presented in Fig. 3 and uses the following notations. $\boldsymbol{v}$ denotes a bit vector of size $n-N$ corresponding to a vertex v, where $\boldsymbol{v}[u]=1$ iff u is a neighbor of v. $\boldsymbol{X}$ denotes a bit vector of size $n-N$ corresponding to chromosome X where $\boldsymbol{X}[v]=1$ iff $X \in \mathrm{CH}(v)$. $\boldsymbol{ext}$ and $\boldsymbol{o}$ are two bit vectors of size $n-N$. $\boldsymbol{ext}[u]=1$ iff u is external. $\boldsymbol{o}[u]=1$ iff u is oriented. The score of each vertex is stored in an integer vector $\boldsymbol{score}$. $\bigwedge$, $\bigvee$, $\bigoplus$ and $\neg$ respectively denote the bitwise-AND, bitwise-OR, bitwise-XOR and bitwise-NOT operators.

One of the major differences between this algorithm and the original algorithm [2] is that in some cases our algorithm performs two passes of maximum score search while Bergeron's algorithm performs only one pass.

1. Choose v with maximal score such that $\boldsymbol{ext}[v]=\boldsymbol{o}[v]=1$.
2. Choose X, Y such that $\boldsymbol{X}[v]=\boldsymbol{Y}[v]=1$.
3. $\boldsymbol{S1} \leftarrow \boldsymbol{X} \bigwedge \boldsymbol{Y} \bigwedge \boldsymbol{v} \bigwedge \neg \boldsymbol{o}$
4. Build the vector $\boldsymbol{S}$ as follows.
   $\boldsymbol{S}[w] \leftarrow 1$ if the following conditions hold:
   - $\boldsymbol{S1}[w]=1$ *(conditions 1 and 2)*
   - $(\boldsymbol{w} \bigwedge \boldsymbol{ext}) \bigvee \boldsymbol{v} = \boldsymbol{v}$ *(condition 3)*
   - $(\boldsymbol{v} \bigwedge \neg \boldsymbol{ext}) \bigvee \boldsymbol{w} = \boldsymbol{w}$ *(condition 4)*
   - $(\boldsymbol{v} \bigwedge \boldsymbol{ext} \bigwedge \boldsymbol{o}) \bigvee \boldsymbol{w} = \boldsymbol{w}$ *(condition 5)*
5. If $\boldsymbol{S} \neq 0$ then flip X:
   a. For every u such that $\boldsymbol{X}[u]=1$:
      i. $\boldsymbol{score} \leftarrow \boldsymbol{score} + \boldsymbol{u}$
      ii. $\boldsymbol{u} \leftarrow \boldsymbol{u} \bigoplus \boldsymbol{X}$
      iii. $\boldsymbol{score} \leftarrow \boldsymbol{score} - \boldsymbol{u}$
   b. Choose v such that $\boldsymbol{S}[v]=1$ and $\boldsymbol{score}[v]$ is maximal.

(Perform $\rho(v)$ where v is an oriented external vertex)

6. $\boldsymbol{score} \leftarrow \boldsymbol{score} + \boldsymbol{v}$
7. $\boldsymbol{v}[v]=1$
8. *For every u such that* $\boldsymbol{v}[u]=1$
   a. *If* $\boldsymbol{ext}[u]=1$*: then* $\boldsymbol{score} \leftarrow \boldsymbol{score} + \boldsymbol{u}$
      else: $\boldsymbol{score} \leftarrow \boldsymbol{score} - \boldsymbol{u}$
   b. $\boldsymbol{u}[u]=1$, $\boldsymbol{u} \leftarrow \boldsymbol{u} \bigoplus \boldsymbol{v}$
   c. *If* $\boldsymbol{ext}[u]=0$*:* $\boldsymbol{X}[u]=1$, $\boldsymbol{Y}[u]=1$, $\boldsymbol{ext}[u]=1$
      Else if $\boldsymbol{X}[u]+\boldsymbol{Y}[v]=2$*:* $\boldsymbol{X}[u]=0$, $\boldsymbol{Y}[u]=0$,$\boldsymbol{ext}[u]=0$
      Else if $\boldsymbol{X}[u]=1$*:* $\boldsymbol{X}[u]=0$, $\boldsymbol{Y}[u]=1$.
      Else if $\boldsymbol{Y}[u]=1$*:* $\boldsymbol{Y}[u]=0$, $\boldsymbol{X}[u]=1$.
   d. *If* $\boldsymbol{ext}[u]=1$*:* $\boldsymbol{score} \leftarrow \boldsymbol{score} - \boldsymbol{u}$
      Else: $\boldsymbol{score} \leftarrow \boldsymbol{score} + \boldsymbol{u}$

Fig. 3. A score-based algorithm for performing a safe translocation

4 A Recursive Algorithm

In this section we present a recursive algorithm for SRT when $\mathcal{IN}(A,B)=\emptyset$. This algorithm is similar to an algorithm by Berman and Hannenhali for SBR [4].

Theorem 4. *Let v be an oriented external vertex in H. Let w be a neighbor of v in H. If $w \in \Delta\mathrm{IN}(H,v)$ then $\Delta\mathrm{IN}(H,w) \subset \Delta\mathrm{IN}(H,v)$.*

Proof. Suppose $w \in \Delta\mathrm{IN}(H,v)$. Obviously $\mathrm{CH}(v)=\mathrm{CH}(w)$. Let x be a vertex in H such that $x \notin \Delta\mathrm{IN}(H,v)$. We shall prove that $x \notin \Delta\mathrm{IN}(H,w)$. Let $x=x_0,\ldots,x_k=y$ be a shortest path from x to an external vertex in $H\cdot\rho(v)$. Then in H: x_j is neighbor of v iff x_j is a neighbor of w, for $j=1..k$.

Case 1: w is oriented in H. Then the subgraphs induced by the vertices $\{x_0,\ldots,x_k\}$ in $H\cdot\rho(w)$ $H\cdot\rho(v)$ are the same. Hence in $H\cdot\rho(w)$: y is external and the path in $x=x_0,\ldots,x_k=y$ exists.

Case 2: w is unoriented in H. In $H\cdot\rho(v)$ the vertices in $\{x_0,\ldots,x_{k-1}\}$ are internal and $x_k(=y)$ is external. Therefore $x_j \in \{x_0,\ldots,x_{k-1}\}$ satisfies in H: *(i)* x_j is a neighbor of v iff x_j is external and $\mathrm{CH}(x_j)=\mathrm{CH}(w)$, and *(ii)* x_j is not a neighbor of v iff x_j is internal. Denote by H' the graph obtained from H after flipping one of the chromosomes in $\mathrm{CH}(w)$.

Case 2.a: at least one vertex in $\{x_0,...,x_{k-1}\}$ is a neighbor of v in H. Choose $x_j \in \{x_0,\ldots,x_{k-1}\}$ a neighbor of v in H such that $\{x_0,\ldots,x_{j-1}\}$ are not neighbors of v in H. Then in H the following conditions are satisfied: *(i)* $x_0,\ldots,x_j$ is a path, *(ii)* all the vertices in $\{x_0,\ldots,x_{j-1}\}$ are internal and *(iii)* x_j is external satisfying $\mathrm{CH}(x_j)=\mathrm{CH}(v)$. Therefore in H' the path $x_0,\ldots,x_j$ still exists and none of the vertices in the path is a neighbor of v (equivalently, w). Hence, the path remains intact in $H'\cdot\rho(w)$.

Case 2.b: none of the vertices in $\{x_0,\ldots,x_{k-1}\}$ is a neighbor of v in H. Then the path $x_0,\ldots,x_k$ exists in H'. v is not a neighbor of w in H' hence v remains external in $H'\cdot\rho(w)$. If x_k is a neighbor of v and w in H' then the path $x_0,\ldots,x_k,v$ exists in $H'\cdot\rho(w)$ and hence $x=x_0 \notin \Delta\mathrm{IN}(H,w)$. If x_k is not a neighbor of v and w in H' then x_k is necessarily external in H' (equivalently, H). Thus none of the subgraphs induced by $\{x_0,\ldots,x_k\}$ in H' and $H'\cdot\rho(w)$ are identical. Hence $x=x_0 \notin \Delta\mathrm{IN}(H,w)$. □

Theorem 5. *If H contains an external vertex then there exists an external vertex v such that $\Delta\mathrm{IN}(H,v) \le \frac{n-N}{2}$.*

Proof. Let v be an external vertex and assume $\mathrm{CH}(v)=\{X,Y\}$. Let $V_{XY}=\{u:\mathrm{CH}(u)=\{X,Y\}\}$. Let $O_{XY}\subseteq V_{XY}$ be the set of oriented vertices in V_{XY}. We can assume w.l.o.g. that $|O_{XY}| \ge \frac{|V_{XY}|}{2}$ (otherwise we flip X).

Case 1: there are two vertices, $v_1,v_2 \in O_{XY}$, which are not neighbors. Let $M_1=\Delta\mathrm{IN}(H,v_1)$ and $M_2=\Delta\mathrm{IN}(H,v_2)$. We shall prove that $M_1 \bigcap M_2=\emptyset$, and hence $\min\{|M_1|,|M_2|\} \le \frac{n-N}{2}$. Assume $u \in M_1$ and let $u=u_0,\ldots,u_k=v_1$ be the shortest path from u to v_1 in H. Since v_2 remains intact in $H\cdot\rho(v_1)$ there is no edge from v_2 to any edge in that path. Therefore this path exists in $H\cdot\rho(v_2)$ and hence $u \notin M_2$.

Case 2: the vertices in O_{XY} form a clique. Let v be a vertex with maximal score ($|N_{\rm IN}(v) - N_{\rm EXT}(v)|$). Then by Lemma 4, $O_{XY} \bigcap \Delta{\rm IN}(H, v) = \emptyset$ and hence $|\Delta{\rm IN}(H, v)| \leq |V_{XY} \setminus O_{XY}| \leq \frac{|V_{XY}|}{2} \leq \frac{n-N}{2}$. □

Algorithm. *Find_Safe_Translocation_Recursive*

1. $\pi_A \leftarrow$ a concatenation of the chromosomes in A
2. Choose v from $H = H(A, B, \pi_A)$ according to Theorem 5.
3. If $\Delta{\rm IN}(H, v) \neq \emptyset$:
   a. $M \leftarrow \Delta{\rm IN}(H, v)$
   b. ${\rm Genes}(M) \leftarrow \{i : (i, i+1) \in M\}$
   c. Let A_M (respectively, B_M) be the genome obtained from A (respectively, B) after deleting all the genes that do not appear in ${\rm Genes}(M)$. Remove common adjacencies of A_M and B_M by deleting one of the genes in each adjacency from both A_M and B_M. Relabel the genes in A_M and B_M such that there is a concatenation of the chromosomes in B_M that is identical to the identity permutation.
   d. $v \leftarrow$ *Find_Safe_Translocation_Recursive*(A_M, B_M)
4. Return v

Fig. 4. A recursive algorithm for locating a safe translocation

Figure 4 presents a recursive algorithm for SRT that follows from Theorems 4 and 5. Note that in step 3.*d* the two genomes A_M and B_M must be co-tailed since their cycle graph contains only cycles. We prove below that each call of the algorithm can be implemented in linear time, hence the algorithm is $O(n^2)$.

Computing $\Delta{\rm IN}(H, v)$: We use a linear time algorithm by Bader, Moret and Yan [1] for computing the components of an overlap graph. The input for the algorithm is the permutation $\pi_A \cdot \rho(v)$. The *span* of a component M is an interval of genes $I(M) = [i, j] \subset \pi_A$, where $i = arg\min\{\pi_A^{-1}(i_1), \pi_A^{-1}(i_2) \mid (i_1, i_2) \in M\}$ and $j = arg\max\{\pi_A^{-1}(j_1), \pi_A^{-1}(j_2) \mid (j_1, j_2) \in M\}$. Clearly we can compute the spans of all the components in linear time. A component is internal iff the two endpoints of its span belong to the same chromosome of A.

Implementation of step 2: Align the vertices of $G(A, B)$ according to π_A. For v, a vertex in H, denote by $Left(v)$ and $Right(v)$ the left and right endpoints respectively of its corresponding grey edge. Find two chromosomes X and Y such that there exists an external vertex that overlaps both of them. Suppose X is found to the left of Y in π_A. Flip if necessary chromosome Y in π_A to achieve $|O_{XY}| \geq \frac{|V_{XY}|}{2}$. Suppose $O_{XY} = \{v_1, \ldots, v_k\}$, where $Left(v_j) < Left(v_{j+1})$ for $j = 1..k-1$.

If there exist two subsequent vertices v_j and v_{j+1} such that $Right(v_j) > Right(v_{j+1})$, then we found two edges that do not overlap. The computation of $\Delta{\rm IN}(H, v_j)$ and $\Delta{\rm IN}(H, v_{j+1})$ is as described above. Otherwise, the vertices in O_{XY} form a clique. We calculate the score for all the vertices in O_{XY} in linear time in the following way. Let $\{I_1, \ldots, I_k\}$ be a set of intervals forming a clique.

Let $U = \{J_1, \ldots, J_l\}$ another set of intervals. Let $U(j)$ denote the number of intervals in U which overlap with I_j. There is an algorithm by Kaplan, Shamir and Tarjan [8] that computes $U(j)$, $j = 1..k$ in $O(k + l)$. We use this algorithm twice to compute $|N_{\text{EXT}}(v_j)|$ and $|N_{\text{IN}}(v_j)|$, for $j = 1..k$.

5 Summary

In spite of the fundamental observation of Hannenhalli and Pevzner that translocations can be mimicked by reversals [6], until recently the analyses of SRT and SBR had little in common. Here and in [12] we tighten the connection between the two problems, by presenting a new framework for the study of SRT that builds directly on ideas and theory developed for SBR. Using this framework we show here how to transform two central algorithms for SBR, Bergeron's score-based algorithm and the Berman-Hannenhalli's recursive algorithm, into algorithms for SRT. These new algorithms for SRT maintain the time complexity of the original algorithms for SBR. These results strengthen our understanding of the connection between the two problems. Still, deeper investigation into the relation between SRT and SBR is needed. In particular, providing a reduction from SRT to SBR or vice versa is an open interesting problem.

Any algorithm that solves SRT can be applied only to genomes that have the same set of tails. In a future work we intend to study an extension of SRT that also allows for non-reciprocal translocations, fissions and fusions and does not require the genomes to be co-tailed.

References

1. D.A. Bader, B. M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.
2. A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.
3. A. Bergeron, J. Mixtacki, and J. Stoye. On sorting by translocations. *Journal of Computational Biology*, 13(2):567–578, 2006.
4. P. Berman and S. Hannenhalli. Fast sorting by reversal. In Daniel S. Hirschberg and Eugene W. Myers, editors, *Combinatorial Pattern Matching, 7th Annual Symposium*, volume 1075 of *Lecture Notes in Computer Science*, pages 168–185, Laguna Beach, California, 10-12 June 1996. Springer.
5. S. Hannenhalli. Polynomial algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71:137–151, 1996.
6. S. Hannenhalli and P. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problems). In *36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 581–592, Los Alamitos, 1995. IEEE Computer Society Press.
7. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46:1–27, 1999. (Preliminary version in Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing 1995 (STOC 95), pages 178–189).

8. H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal of Computing*, 29(3):880–892, 2000. (Preliminary version in Proceedings of the eighth annual ACM-SIAM Symposium on Discrete Algorithms 1997 (SODA 97), ACM Press, pages 344–351).
9. H. Kaplan and E. Verbin. Sorting signed permutations by reversals, revisited. *Journal of Computer and System Sciences*, 70(3):321–341, 2005. A preliminary version appeared in *Proc. CPM03*, Springer, 2003, pages 170–185.
10. J. D. Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the 6th Annual Symposium on Discrete Algorithms*, pages 604–613, New York, NY, USA, January 1995. ACM Press.
11. J. H. Nadeau and B. A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proc Natl Acad Sci U S A*, 81(3):814–818, 1984.
12. M. Ozery-Flato and R. Shamir. An $O(n^{3/2}\sqrt{\log(n)})$ algorithm for sorting by reciprocal translocations. Accepted to CPM 2006. Available at http://www.cs.tau.ac.il/∼ozery/srt_cpm06.pdf.
13. E. Tannier, A. Bergeron, and M. Sagot. Advances on sorting by reversals. *to appear in Discrete Applied Mathematics*.

Inferring Gene Orders from Gene Maps Using the Breakpoint Distance

Guillaume Blin[1], Eric Blais[2], Pierre Guillon[1],
Mathieu Blanchette[2], and Nadia El-Mabrouk[3]

[1] IGM-LabInfo - UMR CNRS 8049 - Université de Marne-la-Vallée - France
{gblin, pguillon}@univ-mlv.fr
[2] McGill Centre for Bioinformatics - McGill University - H3A 2B4 -Canada
{eblais, blanchem}@mcb.mcgill.ca
[3] DIRO - Université de Montréal - H3C 3J7 - Canada
mabrouk@iro.umontreal.ca

Abstract. Preliminary to most comparative genomics studies is the annotation of chromosomes as ordered sequences of genes. Unfortunately, different genetic mapping techniques usually give rise to different maps with unequal gene content, and often containing sets of unordered neighboring genes. Only partial orders can thus be obtained from combining such maps. However, once a total order O is known for a given genome, it can be used as a reference to order genes of a closely related species characterized by a partial order P. In this paper, the problem is to find a linearization of P that is as close as possible to O in term of the breakpoint distance. We first prove an NP-complete complexity result for this problem. We then give a dynamic programming algorithm whose running time is exponential for general partial orders, but polynomial when the partial order is derived from a bounded number of genetic maps. A time-efficient greedy heuristic is then given for the general case, with a performance higher than 90% on simulated data. Applications to the analysis of grass genomes are presented.

1 Introduction

Despite the increase in the number of sequencing projects, the choice of candidates for complete genome sequencing is usually limited to a few model organisms and species with major economical impact. For example, the rice genome is the only crop genome that has been completely sequenced. Other grasses of major agricultural importance such as maize and wheat are unlikely to be sequenced in the short term, due to their large size and highly repetitive composition. In this case, all we have are partial maps produced by recombination analysis, physical imaging and other mapping techniques that are inevitably missing some genes (or other markers) and fail to resolve the ordering of some sets of neighboring genes. Only partial orders can thus be obtained from combining such maps. The question is then to find an appropriate order for the unresolved sets of genes. This is important not only for genome annotation, but also for the study of evolutionary relationships between species. Once total orders have been identified,

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 99–112, 2006.

the classical genome rearrangement approaches can be used to infer divergence histories in terms of global mutations such as reversals [2,9,14].

In a recent study [16,18], Sankoff *et. al.* generalized the rearrangement by reversal problem to handle two partial orders. The idea was to resolve the partial orders into two total orders having the minimal reversal distance with respect to each other. The problem has been conjectured NP-hard, and a branch-and-bound algorithm has been developed for this purpose. The difficulty of this approach is partly due to the fact that both compared genomes have partially resolved gene orders. However, once a total order is known for a given genome, it can be used as a reference to order markers of closely related species. For example, as the grass genomes maintain a high level of conserved synteny [11,7], maps of the completely sequenced rice genome can be used to deduce an order of markers in other grass genomes such as maize.

In this paper, given a reference genome characterized by a total order O and a related genome characterized by a partial order P, the problem is to find a total order coherent with P minimizing the breakpoint distance with respect to O. The underlying criterion is a parsimony one assuming a minimum number of genomic rearrangements. After introducing the basic concepts in Section 2, we show in Section 3 that the considered problem is NP-complete. We then give, in Section 4, two dynamic programming algorithms. First is an algorithm that solves exactly the problem on arbitrary partial orders, and whose worst-case running time is exponential in the number of genes. However, when the partial order considered is the intersection of a bounded number of genetic maps of bounded width, the algorithm runs in polynomial time. We then present a fast and accurate heuristic for the general problem in Section 5. We finally report results on simulated data, and applications to grass genetic maps in Section 6.

2 A Graph Representation of Gene Maps

Hereafter, we refer to elementary units of a map as genes, although they could in reality be any kind of markers. Moreover, as the transcriptional orientation of genes is usually missing from genetic maps, we consider unsigned genes.

A *genetic map* is represented as an ordered sequence of gene subsets or *blocks* $B_1, B_2, \ldots, B_q$, where for each $1 \leq i \leq q$, genes belonging to block B_i are incomparable among themselves, but precede those in blocks $B_{i+1}, \ldots, B_q$ and succeed those in blocks $B_1, \ldots, B_{i-1}$. For example, in Figure 1.a, $\{4, 5\}$ is a block, meaning that genes 4 and 5 are assigned to the same position, possibly due to lack of recombination between them. A genetic map is thus a partial order of genes.

Maps $M_1, \ldots, M_m$ obtained from various protocols can be combined to form a more complex partial order P on the union set of genes of all maps as follows: a gene a precedes a gene b in P if there exists a map M_i where a precedes b. However, combining maps can be a problem in itself, due to possible inconsistencies, which would create precedence cycles (e.g. a precedes b in M_1 but b precedes a in M_2). Breaking cycles can be done in different ways, the parsimonious

method consisting in eliminating a minimum number of precedence rules. Another potential problem is the presence of multiple loci (markers that are assigned to different positions in the same map). These issues have been considered in previous studies [17,18,16], and a software is available for combining genetic maps [10]. In this paper, we assume that the partial order P is already known.

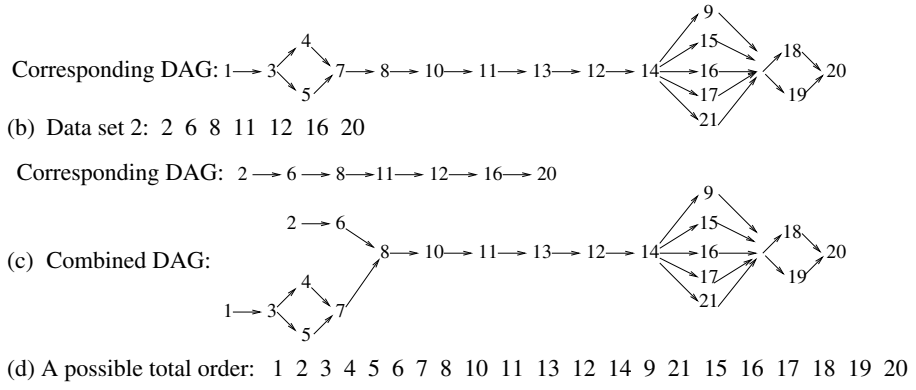


Fig. 1. Data extracted from the comparison of maize and sorghum in the Gramene database. The identity permutation $(1\ 2\ 3\ 4\ \cdots)$ represents the order of markers in the "IBM2 neighbors 2004"[15] map for maize chromosome 5. The corresponding marker's partial orders in sorghum are deduced from (a) "Paterson 2003" [4] map of the chromosome labelled C and (b) "Klein 2004" [13] map of the chromosome labelled LG-01; (c) is the partial order obtained by combining (a) and (b); (d) is a linearization of (c) minimizing the number of breakpoints.

As proposed in previous studies [17,18], we represent such a partial order P as a directed acyclic graph (or DAG) (V_P, E_P), where the vertex set V_P represents the set of genes along the chromosome and the edge set E_P represents the available order information (Figure 1). We consider a minimum set of edges, in the sense that any edge of E_P can not be deduced by transitivity from other edges. In particular, a total order of genes is represented by a DAG such that each edge connects a pair of consecutive genes.

Let P be a partial order represented by a DAG (V_P, E_P). We say that a vertex a is *P-adjacent* to a vertex b and write $a <_P b$ iff there is an edge in E_P from a to b. We say that *a precedes b* in P and write $a \ll_P b$ iff there is a path from a to b. We say that vertices a and b are *incomparable* if neither $a \ll_P b$ nor $b \ll_P a$, and denote this by $a \sim_P b$. A *linearization* of P is a total order O' on the same set of genes, such that $a \ll_P b \Rightarrow a \ll_{O'} b$.

Given a partial order P and a total order O, our goal is to find a linearization O' of P, in such a way that O' is as "similar" as possible to O. The distance measure used here is the number $bkpts(O, O')$ of breakpoints between O and O', where a *breakpoint* is a pair of consecutive vertices (a, b) of O' that are not

consecutive in O ($a <_{O'} b$ but $a \not<_O b$). Equivalently, we may try to maximize the number of O-adjacencies of O', which is defined as the number of pairs of consecutive genes in O' that are also consecutive in O (Figure 1.d). Formally:

Minimum-Breakpoint Linearization (MBL) problem
Given: A partial order P and a total order O on the set of genes $\{1, 2, \ldots, n\}$,
Find: A linearization of P into a total order O' so that $bkpts(O, O')$ is minimized.

Without loss of generality, we assume from now on that O is the identity permutation $(1, 2, \ldots, n)$, that is $i <_O j \Leftrightarrow j = i + 1$.

3 Hardness Results

In this section, we prove that the decision version of the MBL problem is **NP**-complete: given a complete order O and a partial order P defined on the same set of genes and an integer k', can one find a linearization O' of P such that $bkpts(O, O') \leq k'$?

We propose a reduction from the **NP**-complete problem Maximum Independent Set [8]: given a graph $G = (V, E)$ and an integer k, can one find an independent set of vertices of G – *i.e.* a set $V' \subseteq V$ such that no two vertices of V' are connected by an edge in E – of cardinality greater than or equal to k ?

We initially note that the MBL problem is in NP since given a complete order O and a linearization O' of P, one can compute the number of breakpoints in linear time. In order to prove that the MBL problem is **NP**-complete, we show that from any instance of Maximum Independent Set with a parameter k, we are able to construct – in polynomial time – an instance of the MBL problem such that k' depends on k. We detail this construction hereafter.

For convenience, we define a reduction from a slightly different set of instances for the Maximum Independent Set problem: connected graphs. This can be done w.l.o.g. since the problem is still **NP**-complete in that case. Let $G = (V, E)$ be a connected graph of n vertices. We define the complete and the partial orders O and P of the MBL problem as follows. The complete order O is defined as a string $O = \delta\ \alpha_1\ \beta_1\ \gamma_1\ \alpha_2\ \beta_2\ \gamma_2\ \ldots\ \alpha_n\ \beta_n\ \gamma_n\ \epsilon$, and the partial order P as a DAG $P = (V_P, E_P)$ with $V_P = \{\delta, \alpha_1, \alpha_2 \ldots \alpha_n, \beta_1, \beta_2 \ldots\ \beta_n, \gamma_1, \gamma_2, \ldots \gamma_n, \epsilon\}$ and $E_P = \{(\delta, \gamma_1)\} \cup \{(\gamma_i, \gamma_{i+1}) | 1 \leq i < n\} \cup \{(\gamma_n, \alpha_i), (\gamma_n, \beta_i) | 1 \leq i \leq n\} \cup \{(\beta_i, \alpha_j), (\beta_j, \alpha_i) | \forall (v_i, v_j) \in E\} \cup \{(\alpha_i, \epsilon), (\beta_i, \epsilon) | 1 \leq i \leq n\}$.

In order to complete the instance of the MBL problem, we define $k' = (3n + 1) - k$. In the following, we will refer to any such construction as a *MBL-construction*.

First, let us present some interesting properties of any instance of Minimum-Breakpoint Linearization problem obtained by a MBL-construction. Then, we will use those properties to prove that the Minimum-Breakpoint Linearization problem is **NP**-complete. An illustration of a MBL-construction of a graph G of 6 vertices is illustrated in Figure 2.

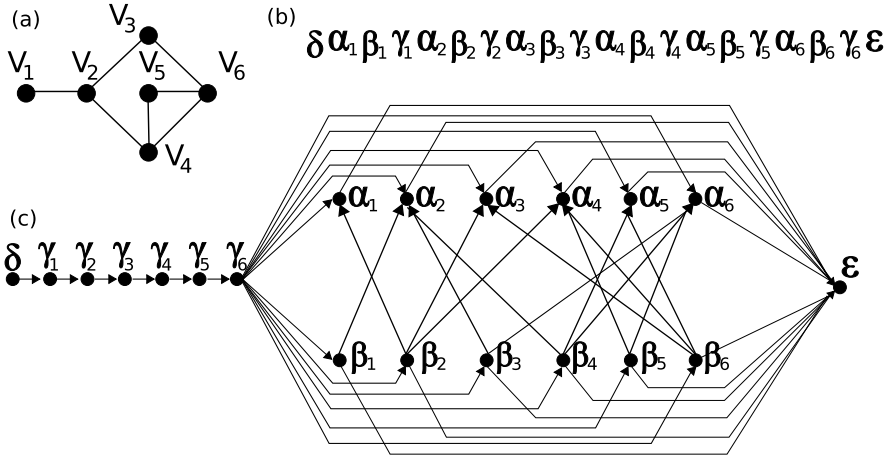


Fig. 2. Example of a MBL-constrction. The graph (a) is a connected graph of 6 vertices. The sequence (b) represents the complete order O and the graph (c) represents the partial order P obtained from the graph (a) by a MBL-construction.

Lemma 1. *Let $G = (V, E)$ be a graph and $P = (V_P, E_P)$ be a partial order obtained from G by a MBL-construction. There exists no linearization O' of P where both $\alpha_i <_{O'} \beta_i$ and $\alpha_j <_{O'} \beta_j$, for any $\alpha_i, \alpha_j, \beta_i, \beta_j$ of V_P such that $(v_i, v_j) \in E$.*

Proof. By contradiction, let us assume that there exists such a linearization O' where $\alpha_i <_{O'} \beta_i$ and $\alpha_j <_{O'} \beta_j$. Since $(v_i, v_j) \in E$, we have $(\beta_i, \alpha_j) \in E_P$ and $(\beta_j, \alpha_i) \in E_P$. Therefore, in any linearization of P – and consequently O' – $\beta_i \ll_{O'} \alpha_j$ and $\beta_j \ll_{O'} \alpha_i$ – which leads, by transitivity, to $\beta_i \ll_{O'} \alpha_i$; a contradiction. □

Lemma 2. *Let $G = (V, E)$ be a graph of n vertices, O and $P = (V_P, E_P)$ be respectively a complete and a partial order obtained from G by a MBL-construction. Given any linearization O' of P, $bkpts(O, O') = (3n+1) - k$ where k is the number of couples (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$.*

Proof. By construction, in O, (i) $\delta <_O \alpha_1 <_O \beta_1 <_O \gamma_1$, (ii) $\forall 1 < i \leq n$, $\gamma_{i-1} <_O \alpha_i <_O \beta_i <_O \gamma_i$ and (iii) $\gamma_n <_O \epsilon$. In any linearization O' of P, (i) $\delta <_{O'} \gamma_1$, (ii) $\forall 1 < i \leq n$, $\gamma_{i-1} <_{O'} \gamma_i$ and (iii) either $\alpha_i <_{O'} \epsilon$ or $\beta_i <_{O'} \epsilon$ for a given $1 \leq i \leq n$. Therefore, in any linearization O' of P, the only adjacencies that can be preserved are the ones of the form $\alpha_i <_O \beta_i$ for some $1 \leq i \leq n$. Let k be the number of couples (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$. If $k = 0$ then no adjacencies at all are preserved, therefore $bkpts(O, O') = (3n+1)$. And consequently, if $k > 0$ then $bkpts(O, O') = (3n + 1) - k$. □

We now turn to prove the following theorem.

Theorem 1. *A connected graph $G = (V, E)$ admits an independent set of vertices $V' \subseteq V$ of cardinality greater than or equal to k if and only if there exists a linearization O' of P such that $bkpts(O, O') \leq (3n + 1) - k$, where O and P result from a MBL-construction of G.*

Proof. ($\Rightarrow$) Let $V' \subseteq V$ such that $|V'| \geq k$ and V' is an independent set. Let O' be a linearization of P defined by $O' = \delta\ P_1\ P_2\ P_3\ P_4\ \epsilon$ where:

- P_1 is the linearization of the subset of vertices $V'_1 = \{\gamma_i | 1 \leq i \leq n\}$ such that $\forall 1 < i \leq n$, $\gamma_{i-1} <_{P_1} \gamma_i$;
- P_2 is **any** linearization of the subset of vertices $V'_2 = \{\beta_i | v_i \in V - V'\}$;
- P_3 is **any** linearization of the subset of vertices $V'_3 = \{\alpha_i, \beta_i | v_i \in V'\}$ such that $\forall v_i \in V'$, $\alpha_i <_{P_3} \beta_i$;
- P_4 is **any** linearization of the subset of vertices $V'_4 = \{\alpha_i | v_i \in V - V'\}$.

By Lemma 2, we can affirm that $bkpts(O, O') = (3n + 1) - |V'|$. Since, by hypothesis, $|V'| \geq k$, we obtain $bkpts(O, O') \leq (3n + 1) - k$.

($\Leftarrow$) Suppose we have a linearization O' of P such that $bkpts(O, O') \leq (3n + 1) - k$. Let $V' \subseteq V$ be the set of vertices such that:

$\forall(\alpha_i, \beta_i)$ such that $\alpha_i <_{O'} \beta_i$, add v_i to V'

By Lemma 1, we can affirm that V' is an independent set. Let us verify that $|V'| \geq k$. By Lemma 2, $Bkpts(O, O') = (3n + 1) - k$ where k is the number of couples (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$. Therefore, we obtain $|V'| = k$. □

4 Exact Dynamic Programming Algorithms

Hereafter, we describe two exact dynamic programming algorithms for solving the MBL problem. The first algorithm works on an arbitrary partial order P, but has a running time that can be exponential in $|V_P|$. However, we show that the algorithm's running time is polynomial in the more realistic case where P is built from a bounded set of genetic maps. The second algorithm applies to the case where P is built from a single genetic map, and runs in linear time.

We begin with some preliminary definitions. Let A be a subset of vertices of V_P. A is a *border* of P iff any pair of vertices of A are incomparable, and a *maximum border* iff any other vertex of V_P is comparable to at least one vertex of A. We also define, for any subset $B \subseteq V_P$, $front(B) = \{x \in B : x \text{ has no successor in } B\}$. Note that the front of any set B is a border. Finally, we denote $pred(A) = A \cup \{x \in V_P : \exists y \in A \text{ s.t. } x \ll_P y\}$.

4.1 A Dynamic Algorithm for Arbitrary Partial Orders

Let A be a maximum border. We denote by $X_{A,i}$ the maximum number of adjacencies that can be obtained from a linearization of $pred(A)$ that is consistent with the partial order P, and that ends with vertex i (*i.e.* i is the rightmost vertex in the total order of $pred(A)$). It is easy to see that the number of adjacencies in the global optimal solution is $\max_{i \in F} X_{F,i}$ adjacencies, where $F = front(V_P)$. The following theorem provides a recursive formula for the computation of $X_{A,i}$.

Theorem 2. *For any border A and any vertex $i \in A$,*

$$X_{A,i} = \max_{j \in A'} X_{A',j} + \begin{cases} 1 & \text{if } |j-i| = 1 \\ 0 & \text{otherwise} \end{cases}$$

where

$$A' = front(pred(A) \setminus \{i\})) = (A \setminus \{i\}) \cup \{k \,|\, (k,i) \in E_P \text{ and } k \notin pred(A \setminus \{i\})\}$$

begins with $A = F = border(V_P)$ and stops as soon as A is the empty set.

Computing all the entries of the dynamic programming table only requires operations which can be done in linear time. If the partial order P admits $b(P)$ possible borders, the running time is $O(b(P) \cdot |V_P|^2)$.

In the general case, the number of borders of P can be as much as $2^{|V_P|}$, if P consists of a single block of incomparable vertices. However, we are more interested in the case where P is obtained by combining m genetic maps, where each map contains a maximum of q blocks and the size of each block is at most k. In this case, there are at most q^m maximal borders in P. Furthermore, two elements that are in the same border cannot be in different blocks on a genetic map, so each maximal border is of size at most km, which allows 2^{km} possible subsets. Therefore, the total number of borders of P is bounded above by $b(P) \in O(q^m 2^{km})$. Since, in practice, only two or three different genetic maps are combined to form a partial order, the dynamic algorithm yields a practical and exact solution to the MBL problem.

4.2 A Linear-Time Algorithm for Single Genetic Map

When P is a genetic map consisting of a list of blocks $B_1, B_2, \ldots, B_q$, a much faster linearization algorithm exists. Let X_i be the maximum linearization score obtained in the partial subset $B_1 \cup \cdots \cup B_i \subseteq V_P$. The maximum linearization score of P is thus equal to X_q. Let L_i represent the set of elements in B_i that can be placed at the last position in a total ordering of $B_1 \cup \cdots \cup B_i$ that achieves the score X_i. Define the functions $g_1(X,Y) = \{x \,|\, x \in X \text{ and } x+1 \in Y\}$ and $g_2(X,Y) = \{y \,|\, y \in Y \text{ and } y-1 \in X\}$. Then, the values X_i and L_i can be determined recursively as follows.

Theorem 3. *Define $X_0 = 0$ and $L_0 = \{\}$. Then, for any $1 \leq i \leq q$,*

$$X_i = X_{i-1} + |g_1(B_i, B_i)| + \begin{cases} 1 \text{ , } \textit{if } |g_2(L_{i-1}, B_i)| \geq 1 \\ 0 \text{ , } \textit{otherwise} \end{cases}$$

and

$$L_i = \begin{cases} B_i \setminus g_1(B_i, B_i) & \text{, if } |g_2(L_{i-1}, B_i)| \neq 1 \text{ or } |B_i| = 1 \\ B_i \setminus (g_1(B_i, B_i) \cup g_2(L_{i-1}, B_i)) & \text{, otherwise} \end{cases}$$

The intuition behind the recursive definition of X_i is as follows: to get the maximum linearization score, we always want to join as many elements $x, x+1$

within a same block. Furthermore, as much as possible, we want to join consecutive elements in neighboring blocks as well. The set L_i is used to keep track of which elements can be put last in the ordering of B_i and therefore possibly be matched with an element in the block B_{i+1}. If the elements of B_i are stored in an ordered list, then the recursive definition of Theorem 3 can be implemented in a recursive algorithm for which each iteration requires $O(|B_i| + |L_{i-1}|)$ time to run, for a total time complexity of $O(n)$ in the case of a genetic map of n genes.

5 An Efficient Heuristic

Since our exact dynamic programming has a worst-case running time that is exponential in the number of genes, a faster heuristic is required to solve large problem instances. In this section, a greedy heuristic is developed for general partial orders obtained from the concatenation of an arbitrary number of maps. It aims to find a maximum number of O-adjacencies coherent with a partial order P. At each step, the partial order is updated by incorporating adjacencies of the longest O-adjacency path that can be part of a linearization of P. The algorithm does not necessarily end up with a total order. Rather, it stops as soon as no more adjacencies can be found. All linearizations of the obtained partial order are then equivalent in the sense that they all give rise to the same number of adjacencies.

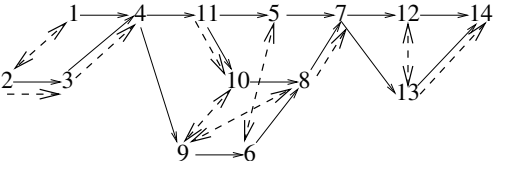


Fig. 3. Dotted edges are all O-adjacencies that can, individually, be part of a linearization of P. A bi-directional edge represents the concatenation of two edges, one in each direction. An adjacency path of P is a directed sequence of consecutive dotted edges.

A *direct* (resp. *indirect*) *adjacency path* of P is a sequence of vertices of form $(i, i+1, i+2, \cdots i+k)$ (resp. $(i+k, \cdots i+2, i+1, i)$) such that for any $0 \leq j < k$, either $i + j <_P i + j + 1$ (resp. $i + j + 1 <_P i + j$), or $i + j$ and $i + j + 1$ are incomparable. For example, in Figure 3, $(1, 2, 3, 4)$ (resp. $(11, 10, 9, 8, 7)$) is a direct (resp. indirect) adjacency path. Notice that adjacencies of this indirect path can not belong to any linearization of P, as gene 5 should be located after 11 but before 7.

We say that an adjacency path p of P is *valid* iff there is a linearization O' of P such that p is a subsequence of O'. Lemma 3 gives the conditions for an adjacency path to be valid. We need a preliminary definition.

Definition 1. *Given two vertices i and j, we say that i is* compatible *with j iff the two following conditions hold:*

1. *i and j are either incomparable or $i \ll_P j$;*
2. *Any vertex v verifying $i \ll_P v \ll_P j$ belongs to the interval $[i, j]$ (or $[j, i]$ if $j < i$).*

Lemma 3. *A direct (resp. indirect) adjacency path of P from i to $i + k$ (resp. from $i + k$ to i) is valid if and only if, for any j_1, j_2 such that $0 \leq j_1 < j_2 \leq k$, $i + j_1$ is compatible with $i + j_2$. (resp. $i + j_2$ is compatible with $i + j_1$).*

Algorithm Find-Valid-Path (P)
{*Compute the list L of all adjacency paths of size 2*}
For $1 \leq i \leq |V|$ *do*
  If ($i <_P i + 1$) or (i and $i + 1$ are incomparable) *then*
    Add $(i, i + 1)$ to L;
End For
$k = 2$;

{*As long as L contains at least two elements, concatenate paths of size k to paths of size k + 1*}
While $|L| \geq 2$ *do*
  For $j = 1$ to $|L|$ *do*
    If L_{j+1} and L_j are consecutive paths *then*
      If $L_j[1]$ is compatible with $L_{j+1}[k]$ *then*
        $L' =$ Concatenate(L_j, L_{j+1});
        Add L' to LNew;
  End For
  If $|LNew| > 0$ *then* $L = LNew$; Clear(LNew);
  $k = k + 1$;
End While
Return (L_1);

Fig. 4. Finding a longest valid adjacency path of P. L is the list of adjacency paths of size k, L_j denotes the j^{th} path of L, and $L_j[i]$ the i^{th} vertex of L_j.

A preliminary preprocessing of $P = (V_P, E_P)$ is required to efficiently compute successive adjacency paths.

1. Create the matrix M of size $|V_P| \times |V_P|$ verifying, for any $i, j \in V_P$, $M(i, j) = 1$ iff $i <_P j$ and $M(i, j) = 0$ otherwise.
2. Compute the transitive closure of M, that is the matrice M^T of size $|V_P| \times |V_P|$ verifying, for any $i, j \in V_P$,

$$M^T(i,j) = \begin{cases} 1 & \text{iff } i <_P j \\ 2 & \text{iff } i \ll_P j \text{ but } i \not<_P j \\ 0 & otherwise \end{cases}$$

After the preprocessing step, the following Steps 1 and 2 are iterated as long as P contains an adjacency path.

- **Step 1:** Find a longest valid direct or indirect adjacency path. An algorithm for this step is described in Figure 4.
- **Step 2:** Incorporate the new adjacencies in M^T, and compute the transitive closure of M^T.

Algorithm Find-Valid-Path (Figure 4) only considers the case of direct paths, though generalization to indirect paths is straightforward. Valid paths are computed beginning with paths of size 2. For a fixed k, any path $p = (i, i+1, \cdots i+k)$ of size k is obtained from a concatenation of two valid consecutive paths $p_1 = (i, i+1, \cdots i+k-1)$ and $p_2 = (i+1, i+2, \cdots i+k)$ of size $k-1$. As p_1 and p_2 are valid paths, the path p is valid iff i is compatible with $i+k$.

Complexity: Computing the transitive closure of the adjacency matrix in the preprocessing phase, as well as in Step 2, is done using the Floyd-Warshall algorithm [6] in time complexity $O(n^3)$ where n is the number of vertices of the corresponding graph. As each condition of *Algorithm Find-Valid-Path* can be checked in constant time and L contains at most $|V| - 1$ elements, the time complexity of Step 1 is in $O(n)$. Moreover, Steps 1 and 2 are iterated at most $|V|$ times. Therefore, the worst time complexity of the greedy algorithm is in $O(n^4)$.

6 Experimental Results

We first test the efficiency of the heuristic compared to the dynamic programming algorithm for general partial orders on simulated data, and then illustrate the method on grass maps obtained from Gramene (http://www.gramene.org/).

Simulated data: We simulate DAGs of fixed size n that can be represented as a linear expression involving the operators '$\rightarrow$' and ',' where P-adjacent genes are separated by a '$\rightarrow$' and incomparable genes by a ','. Such a representation is similar to the one used in [12,17]. For example, the DAG in Figure 1.c has the following string representation:

$$\{2 \rightarrow 6, 1 \rightarrow 3 \rightarrow \{4, 5\} \rightarrow 7\} \rightarrow 8 \cdots 14 \rightarrow \{9, 15, 16, 17, 21\} \rightarrow \{18, 19\} \rightarrow 20$$

DAGs are generated according to two parameters: the *order rate* p that determines the number of ',' in the expression, and the *gene distribution rule* q corresponding to the probability of possible O-adjacencies. We simulated twenty different instances for each triplet of parameters (n, p, q) with $k \in \{30, 50, 80, 100\}$, $p \in \{0.7, 0.9\}$ and $q \in \{0.4, 0.6, 0.8\}$. We did not consider p values lower than 0.7, as the dynamic programming algorithm exponential-time prevented us from testing such instances.

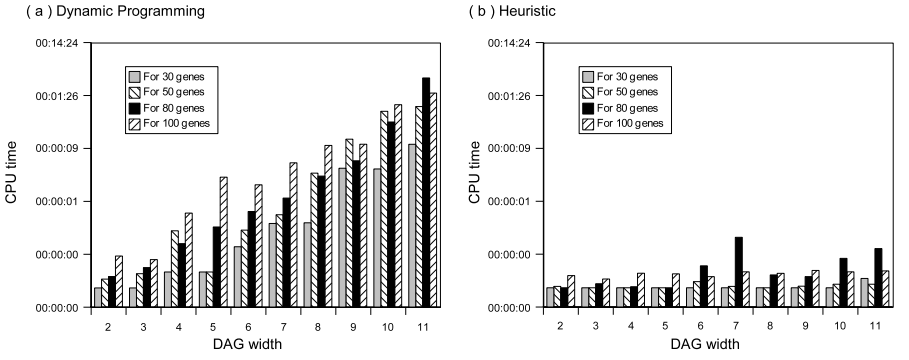


Fig. 5. CPU time expended by (a) the dynamic programming algorithm and (b) the heuristic, for DAGs of a given size and width. Each result is obtained from 10 runs (10 different simulated DAGs). The Y axis is logarithmic.

Figure 5 shows that the running time of the dynamic programming algorithm grows exponentially with the DAG's largest border size, while the heuristic is not affected by it (this was expected, as the time-complexity only depends on the DAG's size). We note that the greedy heuristic can easily handle partial orders consisting of thousands of genes.

We now evaluate the heuristic's optimality, namely the number of O-adjacencies resulting from the obtained linearization compared to the optimal solution (obtained with the dynamic programming algorithm). As illustrated by Table 1, the performance of the greedy algorithm is almost always higher than 90%, and usually close to 100%.

Table 1. Percentage of O-adjacencies resulting from the heuristic's linearization compared to the optimal solution (obtained with the dynamic programming algorithm). Results are obtained by running the heuristic and dynamic programming algorithm on 10 different simulated DAGs for a given size and width.

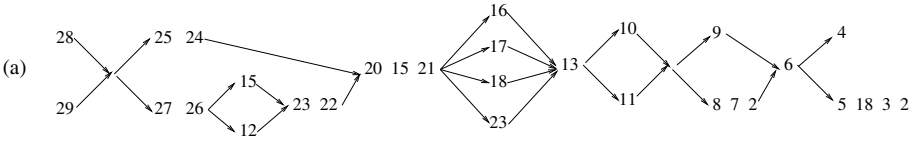
		DAG Width									
		2	3	4	5	6	7	8	9	10	11
Genome size	30	100	100	98,15	97,41	94,33	96,18	93,18	95,54	100	100
	50	100	98,04	96,43	97,62	95,25	98,61	100	86,96	95,26	94,94
	80	100	98,21	97,90	87,54	96,79	93,89	100	95,83	98,33	100
	100	100	98,81	95,65	96,83	89,70	93,95	90,38	95,30	94,63	94,95

Illustration on grass genomes: The Gramene database contains a large variety of maps of different grass genomes such as rice, maize and oats, and provides tools for comparing individual maps. A visualization tool allows to identify regions of 'homeology' between species, that is a linear series of markers in one genome that maps to a similar series of loci on another genome. Integrating

marker orders between different studies remains a challenge to geneticists. However, as total orders are already obtained for widely studied species such as rice which has been completely sequenced, one can use this information to order markers on another species by using the adjacency maximization criterion.

Extracting the linear orders of markers using the Gramene visualization tool remains unpractical for hundreds of markers, as no automatic tool is provided for this purpose. We therefore illustrate the method on maps that are small enough to be extracted manually. Maize has been chosen instead of rice as it has shorter maps, though non-trivial, that can be represented graphically.

We used the "IBM2 Neighbors 2004" [15] map for chromosomes 5 (Figure 1) and 1 (Figure 6) of maize as a reference, and compared it with the "Paterson 2003" [4] and "Klein 2004" [13] maps of the chromosomes labeled C and LG-01, respectively, of sorghum. We extracted all markers of maize indicated as having a homolog in one of the databases of sorghum. All are found completely ordered in maize. This linear order is considered as the identity permutation. For markers of sorghum that are located on maize chromosome 5 (resp. 1), a total order maximizing the adjacency criterion is indicated in Figure 1.d (resp. Figure 6.b).



(b) 29 28 27 26 25 24 15 12 23 22 20 15 21 16 17 18 23 13 11 10 9 8 7 2 6 5 4 18 3 2

Fig. 6. (a) The partial order of markers in sorghum that are located and totally ordered on the maize chromosome 1; (b) A total order maximizing the adjacency criterion

7 Conclusion

We have presented a detailed complexity result and algorithmic study for the problem of linearizing a partial order that is as close as possible to a given total order, in term of the breakpoint distance. Applications on the grass genomes show that this may be helpful to order unresolved sets of markers of some species using the totally ordered maps of well studies species such as rice. However, preliminary to the application of our algorithms is generating the appropriate partial orders. For this purpose, an automated preprocessing of the Gramene comparative database would be required to output the considered genetic maps, and then combine them on a single partial order. The absence of such tools prevented us from presenting more consequent applications.

The next step of this work will be to generalize our approach to two (or more) partial orders, as previously considered in [16,18] for the reversal distance. As conjectured by Sankoff, an NP-complete result for this problem should be proved. A dynaming programming approach may also be envisaged for this case.

Considering the breakpoint (or similarly the adjacency) distance is a first step towards more general distances such as the number of conserved or common intervals [1,3,5]. Indeed, an adjacency of two genes is just a common interval of size 2. A simple extension of the greedy heuristic would be to order genes that remain unordered after maximizing adjacencies, by using the constraint of maximizing intervals of size 3, 4 and so on.

References

1. S. Bérard, A. Bergeron, and C. Chauve. Conservation of combinatorial structures in evolution scenarios. In *LNCS*, volume 3388 of *RECOMB 2004 sat-meeting comp. gen.*, pages 1 - 14. Springer, 2004.
2. A. Bergeron, J. Mixtacki, and J. Stoye. Reversal distance without hurdles and fortresses. volume 3109 of *LNCS*, pages 388 - 399. Springer-Verlag, 2004.
3. G. Blin and R. Rizzi. Conserved interval distance computation between non-trivial genomes. COCOON, 2005.
4. J.E. Bowers, C. Abbey, A. Anderson, C. Chang, X. Draye, A.H. Hoppe, R. Jessup, C. Lemke, J. Lennington, Z.K. Li, Y.R. Lin, S.C. Liu, L.J. Luo, BS. Marler, R.G. Ming, S.E. Mitchell, D. Qiang, K. Reischmann, S.R. Schulze, D.N. Skinner, Y.W. Wang, S. Kresovich, K.F. Schertz, and A.H. Paterson. A high-density genetic recombination map of sequence-tagged sites for Sorghum, as a framework for comparative structural and evolutionary genomics of tropical grains and grasses. *Genetics*, 2003.
5. M. Figeac and J.S. Varré. Sorting by reversals with common intervals. In *LNBI*, volume 3240 of *WABI 2004*, pages 26 - 37. Springer-Verlag, 2004.
6. Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 1962.
7. M.D. Gale and K.M. Devos. Comparative genetics in the grasses. *Proceedings of the National Academy of Sciences USA*, 95:1971- 1974, 1998.
8. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.
9. S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Journal of the ACM*, 48:1–27, 1999.
10. B.N. Jackson, S. Aluru, and P.S. Schnable. Consensus genetic maps: a graph theory approach. In *IEEE Computational Systems Bioinformatics Conference (CSB'05)*, pages 35- 43, 2005.
11. B. Keller and C. Feuillet. Colinearity and gene density in grass genomes. *Trends Plant Sci.*, 5:246- 251, 2000.
12. S.E Lander, P. Green, J. Abrahamson, and A. Barlow amd M.J Daly *et al.* MAPMAKER: an interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics*, 1:174 - 181, 1987.
13. M.A. Menz, R.R. Klein, J.E. Mullet, J.A. Obert, N.C. Unruh, and P.E. Klein. A High-Density Genetic Map of Sorghum Bicolor (L.) Moench Based on 2926 Aflp, Rflp and Ssr Markers. *Plant Molecular Biology*, 2002.
14. P.A. Pevzner and G. Tesler. Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *Proc. Natl. Acad. Sci. USA*, 100:7672 - 7677, 2003.

15. M.L. Polacco and Jr Coe E. IBM neighbors: a consensus GeneticMap. 2002.
16. D. Sankoff, C. Zheng, and A. Lenert. Reversals of fortune. *Proceedings of the 3rd RECOMB Comparative Genomics Satellite Workshop*, 3678:131–141, 2005.
17. I.V. Yap, D. Schneider, J. Kleinberg, D. Matthews, S. Cartinhour, and S. R. McCouch. A graph-theoretic approach to comparing and integrating genetic, physical and sequence-based maps. *Genetics*, 165:2235- 2247, 2003.
18. C. Zheng, Aleksander Lenert, and D. Sankoff. Reversal distance for partially ordered genomes. *Bioinformatics*, 21, 2005. in press.

Ordering Partially Assembled Genomes Using Gene Arrangements

Éric Gaul and Mathieu Blanchette

McGill Centre for Bioinformatics, McGill University
{egaul, blanchem}@mcb.mcgill.ca

Abstract. Several mammalian genomes will only be sequenced at a 2X coverage, resulting in the impossibility of assembling contigs into chromosomes. We introduce the problem of ordering the contigs of two partially assembled genomes so as to maximize the similarity (measured in terms of genome rearrangements) between the assembled genomes. We give a linear-time algorithm for the Block Ordering Problem (BOP): Given two signed permutations (genomes) that are been broken into blocks (contigs), order and orient each set of blocks, in such a way that the number of cycles in the breakpoint graph of the resulting permutations is maximized. We illustrate our algorithm on a set of 90 markers from the human and mouse chromosomes X and show how the size of the contigs and the rearrangement distance between the two genomes affects the accuracy of the predicted assemblies. The appendix and an implementation are available at www.mcb.mcgill.ca/~egaul/recomb2006.

Keywords: Genome rearrangement, gene order,breakpoint graph, genome assembly.

1 Introduction

The completion of several vertebrate genome sequencing projects (human, chimp, mouse, rat, dog, chicken, etc.) has allowed detailed analyses of the processes and history of genome rearrangement during mammalian evolution [1,2,3]. Sophisticated methods now exist to study genome rearrangements on large, multichromosomes genomes [4,5,6,7]. Still, some debate remains regarding the specific arrangement of markers in ancestral genomes [8,9], and regarding the mechanisms at play (e.g. breakpoint re-use [10,11]). One of the best approaches to resolve these questions is to use data from more species, but sequencing complete vertebrate genomes is an expensive feat. While the sequencing of large quantities of short reads is highly automated and relatively cheap, the finishing step, in which larger contigs are ordered and oriented into chromosomes, remains challenging. For this reason, many vertebrate genomes (cow, cat, rabbit, armadillo, elephant, tenrec, etc.) are now being partially sequenced (2X coverage) and the contigs are left unassembled. While this incomplete information is sufficient for many applications (e.g. detection of regions under selective pressure [12,13]), it

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 113–128, 2006.

has, until now, been considered that this data is useless for studying genome rearrangements.

In this paper, we introduce and solve a new algorithmic problem called the BLOCK ORDERING PROBLEM. The input consists of two partially assembled genomes, each represented as an unordered set of blocks (contigs). Each block consists of an ordered list of one or more markers. Intuitively, our goal is to assemble the two genomes, using each other as incomplete reference, in such a way that the rearrangement distance between the two assembled genomes is minimized. Based on a parsimony criterion, such assemblies are those that are the most likely to be correct. To be more precise, our algorithm outputs an assembly of the two genomes that maximizes the number of cycles in the breakpoint graph they induce. Maximizing the number of cycles in a breakpoint graph has been shown to approximate very well the reversal distance between them [14]. This first step in the study of genome rearrangements in the presence of partially assembled genomes opens the door to a number of more difficult combinatorial problems (see Future work and conclusion).

2 The Block Ordering Problem (BOP)

We assume that completely assembled genomes can be represented as signed permutations of numbers between 1 and n, where each marker i is present exactly once (in either orientation) in each of the two genomes considered. In this paper, we focus on uni-chromosomal, linear genomes, but the algorithms are easily adapted to circular genomes. Given two unfragmented permutations π_1 and π_2, the problem of inferring most parsimonious rearrangement scenarios between the two has received much attention over the last ten years and polynomial time algorithms exist for most types of rearrangement operations [15,16,17].

A fragmented genome can be represented as an unordered set of *blocks*. A block B of size s is a non-empty ordered list of s signed integers (markers), denoted $B = [i_1, i_2, ..., i_s]$. Since the orientation of each block is unknown, we also need to consider the *reverse* of each block B, denoted $\overline{B} = [-i_s, \ldots, -i_2, -i_1]$. For convenience, we define $B_0 = [0]$ and $B_{k+1} = [k + 1]$ as "dummy" blocks that mark the beginning and end of the genome. For example, a set of blocks could be $\mathcal{B} = \{[0], [-7, -6, -3, 4], [5, 8, 9], [-2, -1], [10]\}$. An *ordering $P_\mathcal{B}$ of a set of blocks* $\mathcal{B}$ is an ordered list where each block or its reverse appears exactly once, and that starts with B_0 and ends with B_{k+1}. Each ordering of a set of blocks naturally induces a signed permutation $P_\mathcal{B}$ of size n by concatenating the blocks of the list (in the following, we will use $P_\mathcal{B}$ to equally designate the ordering or the induced permutation). For example, $P_\mathcal{B} = [[0], [1, 2], [-9, -8, -5], [-4, 3, 6, 7], [10]]$ is an ordering of $\mathcal{B}$ and induces the permutation (0 1 2 −9 −8 −5 −4 3 6 7 10). This motivates the following optimization problem, which is illustrated in Fig. 1.

BLOCK ORDERING PROBLEM (BOP)
Given: Two sets of blocks $\mathcal{A}$ and $\mathcal{B}$ on n.
Find: An ordering $P_\mathcal{A}$ of $\mathcal{A}$ and an ordering $P_\mathcal{B}$ of $\mathcal{B}$ such that a criterion $c(P_\mathcal{A}, P_\mathcal{B})$ is optimized.

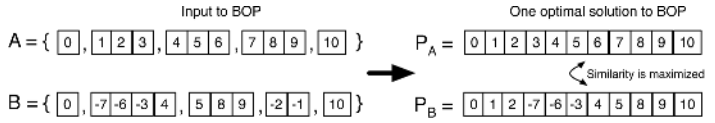


Fig. 1. Example of input to the Block Ordering Problem, together with a solution that maximizes the number of cycles in the breakpoint graph of the resulting orderings (and, in this case, that also minimizes the reversal distance between them).

In this paper, we describe algorithms to solve the BOP in the case where the criterion $c(P_{\mathcal{A}}, P_{\mathcal{B}})$ being maximized is the number of cycles in the breakpoint graph of the two permutations (see details below). This quantity has been shown by Hannenhalli and Pevzner [16] to be a key determinant of the reversal distance between the two permutations.

2.1 Related Work

Although, to our knowledge, partially assembled genomes of the type described in the introduction have never been used in the context of genome rearrangement studies, related questions have been studied in other contexts. A problem similar to the BOP arises when trying to infer a minimal number of reversals and translocations required to transform one multi-chromosome (but fully assembled) genome into a another [7]. The current solution involves capping chromosomes with special markers and then ordering them into a single linear "meta-chromosome" in which reversals can be used to model intra-chromosomal reversals, inter-chromosomal translocations, as well as chromosome fusions and fissions. Although superficially similar, our problem is different in many aspects. Ends of blocks are not similar to ends of chromosomes. The breaking of the permutation into blocks represent missing information about the order of markers, and is an experimental artifact, not the result of an evolutionary process.

Another related problem, introduced by Zheng *et al.* [18], is that of comparing genetic maps represented as partial orders on a set of markers. This problem is very different from the BOP because a block contains possibly many markers which are totally ordered (as a subset), and between which it is not possible to insert any other marker from another block. It is thus impossible to use the formalism and method developed by Zheng *et al.* [18] to solve the BOP.

3 The Breakpoint and Fragmented Breakpoint Graphs

The precise definition of the problem addressed in this paper requires a few definitions. Let $B = [i_1, ..., i_s]$ be a block belonging to the block set $\mathcal{A}$. We say that i_k *precedes* i_{k+1} in $\mathcal{A}$, for all $k = 1...s-1$, and we denote this $i_k \rhd_{\mathcal{A}} i_{k+1}$. We also say that $-i_{k+1}$ precedes $-i_k$ in $\mathcal{A}$. Thus, B and $\overline{B}$ have the same set of pairs for the precedence relation. Each element i_j of B will be flanked by a left point $l(i_j)$ and a right point $r(i_j)$. If $i_j > 0$, we define $l(i_j) = i_j^-$ and $r(i_j) = i_j^+$. If $i_j < 0$, we define $l(i_j) = |i_j|^+$ and $r(i_j) = |i_j|^-$. Each block B has two ends:

$l(i_1)$ and $r(i_s)$. For example, if $B = [3, 4, -5]$, the two ends are 3^- and 5^-. Notice that B and $\overline{B}$ have the same ends. If block sets $\mathcal{A}$ and $\mathcal{B}$ both consist of a single block (i.e. they are permutations), a classic object originally defined by Hannenhalli and Pevzner [16] is the breakpoint graph of $\mathcal{A}$ and $\mathcal{B}$:

Definition 1 (Breakpoint Graph [16]). *Given two sets of blocks on n, $\mathcal{A}$ and $\mathcal{B}$, each consisting of a single block, their* Breakpoint graph $G_{\mathcal{AB}} = (V_{\mathcal{AB}}, E_{\mathcal{AB}})$ *is a the multi-graph defined as follows:*

- $V_{\mathcal{AB}} = \{0^+, 1^-, 1^+, 2^-, 2^+, \ldots, n^-, n^+, (n+1)^-\}$,
- *For every i, j such that $i \rhd_{\mathcal{A}} j$, there is an edge $\{r(i), l(j)\}$ in $E_{\mathcal{AB}}$. Similarly, for every i, j such that $i \rhd_{\mathcal{B}} j$, there is an edge $\{r(i), l(j)\}$ in $E_{\mathcal{AB}}$. Two vertices can thus be connected by up to two edges.*

As mentioned in the Introduction, Hannenhalli and Pevzner [16] have shown that the minimal number of reversals required to transform a permutation into another permutation of the same size (known as the reversal distance between the permutations) is given by $n - c + t$, where n is the size of the permutations, c is the number of cycles in their breakpoint graph $G_{\mathcal{AB}}$ and t is the number of reversals needed to get an "easy" configuration[1]. Since the value of t is usually very small [14], minimizing the reversal distance is thus nearly equivalent to maximizing the number of cycles. In the future, we intend to include t in the optimization process.

Definition 2. *Given two block sets $\mathcal{A}$ and $\mathcal{B}$, the* score $c(P_{\mathcal{A}}, P_{\mathcal{B}})$ *of the simultaneous orderings $(P_{\mathcal{A}}, P_{\mathcal{B}})$ is the number of cycles in the breakpoint graph $G_{P_{\mathcal{A}} P_{\mathcal{B}}}$.*

We clearly cannot build a breakpoint graph for our problem, because we do not know *a priori* a total order for the markers in any of the two data sets. Instead, we define the Fragmented Breakpoint Graph, which is a generalization of the standard breakpoint graph defined above. Refer to Fig. 2 for an example.

Definition 3 (Fragmented Breakpoint Graph). *Given two sets of blocks $\mathcal{A}$ and $\mathcal{B}$ on n, their* Fragmented breakpoint graph $F_{\mathcal{AB}} = (V^F_{\mathcal{AB}}, E^F_{\mathcal{AB}})$ *is the vertex-colored multi-graph where:*

- $V^F_{\mathcal{AB}} = \{0^+, 1^-, 1^+, 2^-, 2^+, \ldots, n^-, n^+, (n+1)^-\}$
- *Vertex $i \in V^F_{\mathcal{AB}}$ is colored* $\begin{cases} \textit{white} & \textit{if } i \textit{ is a block end in } \mathcal{A} \textit{ but not in } \mathcal{B}. \\ \textit{black} & \textit{if } i \textit{ is a block end in } \mathcal{B} \textit{ but not in } \mathcal{A}. \\ \textit{grey} & \textit{if } i \textit{ is a block end in both } \mathcal{A} \textit{ and } \mathcal{B}. \\ \textit{red} & \textit{otherwise.} \end{cases}$
- *As in the breakpoint graph, for every i, j such that $i \rhd_{\mathcal{A}} j$ and for every i, j such that $i \rhd_{\mathcal{B}} j$, there is an edge $\{r(i), l(j)\}$ in $E_{\mathcal{AB}}$.*

[1] The value t is the minimal number of reversals needed to get rid of "bad" (unoriented) components and to get a permutation which can be sorted in $n - c$ reversals [15,19].

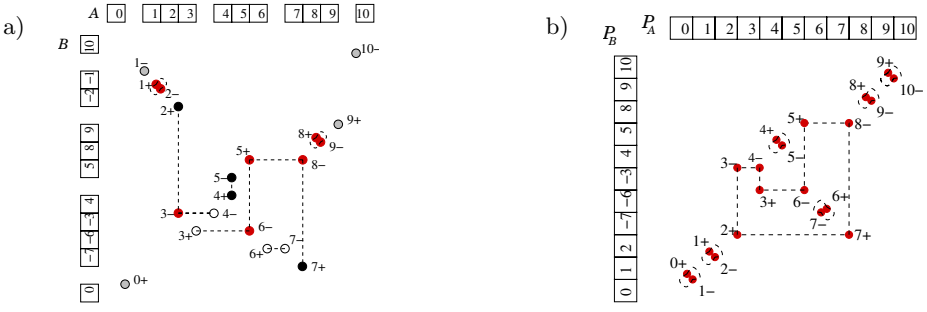


Fig. 2. (a) The fragmented breakpoint graph, laid out as suggested in text, for the two sets of blocks $\mathcal{A} = \{[0], [1, 2, 3], [4, 5, 6], [7, 8, 9], [10]\}$ and $\mathcal{B} = \{[0], [-7, -6, -3, 4], [5, 8, 9], [-2, -1], [10]\}$. (b) The breakpoint graph $G_{P_\mathcal{A} P_\mathcal{B}}$ for the two permutations $P_\mathcal{A} = (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10)$ and $P_\mathcal{B} = (0\ \ 1\ \ 2\ \ -7\ \ -6\ \ -3\ \ 4\ \ 5\ \ 8\ \ 9\ \ 10)$. This is an optimal solution for the block ordering problem for $\mathcal{A}$ and $\mathcal{B}$, for which the number of cycles is 7.

Notice that if $\mathcal{A}$ and $\mathcal{B}$ each consist of a single block, their fragmented breakpoint graph is the same as their breakpoint graph. Following Pevzner and Tesler [20], the fragmented breakpoint graph is drawn by representing block sets as vertical and horizontal axis, and vertices aligned with the element ends they represent in each block set (as in Fig. 2). We observe that the fragmented breakpoint graph is composed exclusively of simple paths and of cycles (an isolated vertex being seen as a trivial path). Moreover, all vertices are red, except those at the start and end of each path. Notice that the fragmented breakpoint graph can be computed from the block sets in $O(n)$, where n is the number of markers.

4 The Block Ordering Graph

Our goal is to find simultaneously an ordering for both sets of blocks so as to maximize the number of cycles in the breakpoint graph resulting from the permutations. Observe that joining two blocks simply adds one edge to the fragmented breakpoint graph. We thus have the following propositions.

Proposition 1. *Joining two blocks from $\mathcal{A}$ or from $\mathcal{B}$ can only add one edge to the graph $F_{\mathcal{AB}}$, between vertices of degree zero or one. This can only result in joining two paths to form a new, longer path, or in closing a path to form a cycle.* □

Proposition 2. *The score for an instance $(\mathcal{A}, \mathcal{B})$ of the BOP cannot exceed the number of connected components of the fragmented breakpoint graph $F_{\mathcal{AB}}$.* □

Proposition 3. *If the fragmented breakpoint graph for $\mathcal{A}$ and $\mathcal{B}$ contains a cycle, then this cycle will be present in the breakpoint graph derived from any orderings $(P_\mathcal{A}, P_\mathcal{B})$.* □

One consequence of Proposition 3 that is that we can ignore in our quest all cycles in $F_{\mathcal{AB}}$, since they will remain unchanged no matter how the blocks are arranged. Our task is thus to close the greatest possible number of paths of the fragmented breakpoint graph (thus forming cycles), but avoiding merging them as much as possible. The rest of this section describes the structure that will allow to do this optimally.

Having the last remarks in mind, we build a simplified version of the fragmented breakpoint graph, that we call the Block Ordering Graph. This graph is obtained by removing unnecessary information from the fragmented breakpoint graph and focusing on information that is relevant to the task. Information representing blocks from the block sets is also added (refer to Fig. 3 for an example).

Definition 4 (Block Ordering Graph). *Given two sets of blocks $\mathcal{A}$ and $\mathcal{B}$, their* Block Ordering Graph*, or* BOG*, is the vertex-bicolored, edge-bicolored graph $H_{\mathcal{AB}} = (V^H_{\mathcal{AB}}, E^H_{\mathcal{AB}})$ where:*

- *Vertices are derived from the fragmented breakpoint graph $F_{\mathcal{AB}}$ according to the following rules:*
  - *if i is a white vertex in $F_{\mathcal{AB}}$, then i is a white vertex in $H_{\mathcal{AB}}$.*
  - *if i is a black vertex in $F_{\mathcal{AB}}$, then i is a black vertex in $H_{\mathcal{AB}}$.*
  - *if i is a gray vertex in $F_{\mathcal{AB}}$, then i is duplicated in a white vertex and a black vertex in $H_{\mathcal{AB}}$.*
  - *if i is a red vertex in $F_{\mathcal{AB}}$, then i is not present in $H_{\mathcal{AB}}$.*
- *Edges are* dashed *or* solid*:*
  - *if vertices i and j form the ends of a path in $F_{\mathcal{AB}}$, then $\{i, j\}$ forms a dashed edge in $H_{\mathcal{AB}}$. There is also a dashed edge between any pair of vertices with the same label but with different colors. Dashed edges joining two vertices of the same color are called* one-sided edges*, while those joining two vertices of different colors are called* two-sided edges*.*
  - *if i and j are of the same color and form the left and right ends of the same block in $\mathcal{A}$ or in $\mathcal{B}$, then $\{i, j\}$ is a solid edge in $H_{\mathcal{AB}}$.*

From the way it is derived from the fragmented breakpoint graph, it is clear that the BOG is composed exclusively of paths and cycles. By looking at a general situation (like the one illustrated in Fig. 3b), we identify different kinds of connected components in the BOG: *one-sided* components are composed exclusively of vertices of a same color, and *two-sided* components contain vertices of both colors. We call *starting* and *ending* components the two single-edge components joining the two 0^+ vertices and the two $(n+1)^-$ vertices, respectively. The following observation will have important consequences for deciding how to order blocks.

Proposition 4. *With the exception of the starting and ending components, all components in the BOG are dashed-solid alternating cycles. Moreover, every two-sided component has an even number of two-sided edges.* □

Note that the block ordering graph can be computed from the fragmented breakpoint graph in linear time (with respect to the number of markers).

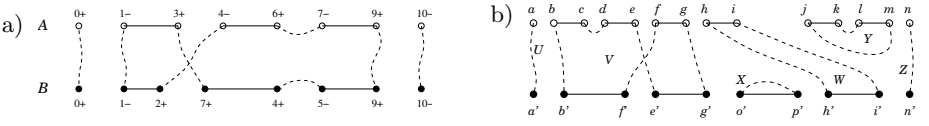


Fig. 3. (a) The BOG derived from the fragmented breakpoint graph of Fig. 2. White vertices represent ends from block set $\mathcal{A}$, and black vertices, those from $\mathcal{B}$. Solid edges join ends of a same block, and dashed edges represent ends that are part of the same path in the fragmented breakpoint graph. $\{6^+, 7^-\}$ is a one-sided edge, while $\{4^-, 2^+\}$ is a two-sided edge. (b) A BOG showing the four possible types of connected components. The starting component is U, the ending component is Z, components X and Y are one-sided, and components U, V, W and Z are two-sided. Notice that α-one-sided edges are part of one-sided components (e.g. $\{k, l\}$), and β-one-sided edges, of two-sided components (e.g. $\{c, d\}$).

5 Determining Optimal Orderings

Recall that each dashed edge in the block ordering graph $H_{\mathcal{AB}}$ corresponds to a path in the fragmented breakpoint graph $F_{\mathcal{AB}}$, and that we would like to close as many of these paths as possible, by joining together pairs of blocks. In this section, we first show how to close the paths corresponding to one-sided dashed edges, and then address the more difficult case of two-sided edges.

5.1 Processing One-Sided Edges of the BOG

There are two types of one-sided edges in the BOG: those that are part of one-sided components, that we will call the *α-one-sided edges*, and those that are part of two-sided components, the *β-one-sided edges*. We start the discussion with the latter type, which is the easiest to process.

β-one-sided edges, the one-sided edges from two-sided components. We observe that one-sided edges involved in two-sided components (the β-one-sided edges) correspond, in the fragmented breakpoint graph, to paths that can be closed by simply joining two blocks in either $\mathcal{A}$ or in $\mathcal{B}$. If it is done, the problem looks the same as before, but some blocks have been merged into longer ones, so we can imagine updating the BOG correspondingly. For example, in Fig. 3b, merging together ends c and d results in a new "block" $[b..e]$. Obviously, every β-one-sided edges can be processed this way before going further. The proposed algorithm (described later) applies this simplification to the BOG, and does the required bookkeeping to output correct orderings at the end.

Remark 1. We assume from now on that all the blocks connected by β-one-sided edges have been joined into a single block, so that all two-sided components consist only of two-sided edges and solid edges.

α-one-sided edges, the one-sided edges from one-sided components. We would like to apply the processing described above to one-sided edges that

are part of one-sided components, the α-one-sided edges. Unfortunately, not all α-one-sided edges can simultaneously be "closed" that way, because this may lead to an invalid solution: since each one-sided components is an alternating cycle in the BOP, closing all its dashed edges would correspond to joining all the corresponding block ends, ultimately resulting into a cycle of blocks. For example, in Fig. 3b, connected component Y, joining together both $\{k, l\}$ and $\{j, m\}$ results in the two blocks forming a cycle, and thus being impossible to include in the global solution. Instead, one of the two dashed edges needs to be sacrificed, resulting in four possible partial ordering: $[[j..k][l..m]]$, $[[-m.. -l][-k.. -j]]$, $[[l..m][j..k]]$ and $[[-k.. -j][-m.. -l]]$. One thus has to choose an edge to sacrifice for each one-sided component. The good news is that any α-one-sided edge can be sacrificed and the resulting partial ordering of blocks can be inserted anywhere in the complete orderings, without changing the score of the solution, according to the following proposition.

Proposition 5. *Let $(P_{\mathcal{A}}, P_{\mathcal{B}})$ be any simultaneous orderings of instance $(\mathcal{A}, \mathcal{B})$ (not necessarily optimal), and let C be a sublist of blocks of $P_{\mathcal{A}}$ (resp. $P_{\mathcal{B}}$) such that the ends i and j of the list are joined by a one-sided edge $e = \{i, j\} \in E^H_{\mathcal{AB}}$. Then moving the sublist C as a whole in the ordering $P_{\mathcal{A}}$ does not change the number of cycles in the breakpoint graph $G_{P_{\mathcal{A}} P_{\mathcal{B}}}$.*

Proof. See Appendix. □

The conclusion to this section is that one-sided edges can be dealt with easily and independently of two-sided edges. We will see later that this way of doing things guarantees optimality of the output.

5.2 Processing Two-Sided Edges of the BOG

We now turn to the problem of dealing with two-sided components in a BOG. We assume that all β-one-sided edges have been dealt with already, so that two-sided components only consist of two-sided edges and solid edges. Closing the paths corresponding to two-sided edges cannot be done by joining two blocks from the same set ($\mathcal{A}$ or $\mathcal{B}$), because these paths have one end in $\mathcal{A}$ and the other end in $\mathcal{B}$. Closing such a path thus requires to merge it with at least one other path. Moreover, we need at least one additional end from each block set to close that path. Since our goal is to maximize the final number of cycles, the best way to choose these two additional free ends is to take them from another two-sided path. Thus, if a BOG contains k two-sided edges[2], the number of cycles resulting from these two-sided edges will be at most $k/2$. We now show that we can actually reach this upper bound by pairing correctly the two-sided edges.

Unfortunately (or fortunately, for the sake of reducing ambiguity), it is not always possible to pair together an arbitrary choice of two-sided edges. One

[2] It follows from Prop. 4 that the number of two-sided edges in any BOG is even.

constraint is that it is impossible to join together a pair of two-sided edges that are connected to a same block (solid edge) (for example: $\{b, b'\}$ and $\{f, f'\}$ in Fig. 3b), because this would imply that the ends of this block are adjacent in the solution, which is not possible. Another problem arises from the requirement that all blocks from each set must appear within the solution. For example, from Fig. 3b, joining block ends a and g, f and i, h and b, and e and n yields in the ordering $[[a], [-g..-f], [-i..-h], [b..e], [n]]$ for the white blocks and $[[a'], [-g'..-e'], [n']]$ for the black blocks. The last of these partial orderings is not valid, because it does not include all the blocks incident to two-sided edges. Fortunately, it is always possible to find a pairing of two-sided edges that yields a valid ordering of the blocks involved in two-sided components, and the rest of this section shows how.

Definition 5 (Edge Matching Graph). *The* Edge Matching Graph $E_{\mathcal{AB}}$ *is the edge-colored graph defined as follows:*

- *Vertices are the two-sided edges from the BOG* $H_{\mathcal{AB}}$*, the* starting (resp. ending) vertex *corresponding to the starting (resp. ending) component of* $H_{\mathcal{AB}}$*;*
- *There is an edge between two vertices* v *and* w *if the corresponding two-sided edges in* $H_{\mathcal{AB}}$ *are the ends of the solid edge* $\{v, w\} \in E^H_{\mathcal{AB}}$*. The edge is white if* v *and* w *are white, and black otherwise. For convenience, there are also a white* and *a black edge between the starting and ending vertices.*

Figure 4a gives an example of the edge matching graph corresponding to the block ordering graph of Fig. 3b. Observe that in the edge matching graph, each connected component is an alternating cycle. The problem of finding a matching of two-sided edges is now restated formally as the following problem in the edge matching graph:

Definition 6 (The Edge Matching Problem). *Given an edge matching graph* $J_{\mathcal{AB}} = (V^J_{\mathcal{AB}}, E^J_{\mathcal{AB}})$*, the* Edge matching Problem *is the problem of adding a third kind of edges (the green edges) such that the following conditions hold:*

1. *The green edges (considered alone) define a perfect matching of the vertices of* $J_{\mathcal{AB}}$*;*
2. *There is an alternating path from the start to the end vertices that visits all green and white edges;*
3. *There is an alternating path from the start to the end vertices that visits all green and black edges.*

A solution to the instance of the problem of Fig. 4a is illustrated in Fig. 4b. The problem is defined as above because a solution to this problem is a legal and optimal matching of two-sided edges in the BOG. The matching condition will merge the paths corresponding to all two-sided edges to get a maximal number of cycles, and the two other conditions ensure that each block from each set is

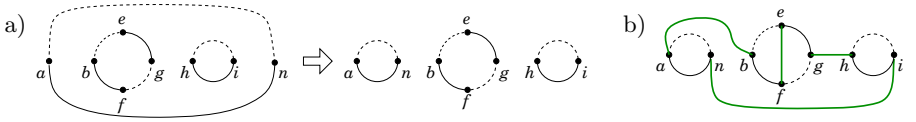


Fig. 4. (a) The edge matching graph for the BOG from Fig. 3b. In the current figure, white edges are dashed and black edges are solid, a is the starting vertex, and n the ending vertex. Notice that vertices of one-sided edge $\{c, d\}$ separating $\{b, c\}$ and $\{d, e\}$ have been joined together, to give birth to the new solid edge $\{b, e\}$ (see Remark 1). (b) One solution for the matching problem of the edge matching graph shown in (a), in which the green edges are in bold. The solution correspond to simultaneous partial orderings $[[a], [b..e], [f..g], [h..i], [n]]$ and $[[a'], [b'..f'], [e'..g'], [h'..i'], [n']]$. These partial orderings contribute four cycles in an eventual complete ordering of blocks.

part of the corresponding ordering[3]. It is relatively easy to find a solution to the edge matching problem (see Algorithm 1).

Definition 7. *Let $J_{\mathcal{AB}}$ be an edge matching graph. A partial matching M of the vertices of $J_{\mathcal{AB}}$ using green edges and such that $|M| < |V^J_{\mathcal{AB}}|/2$ is* valid *if the graph $(V_{\mathcal{AB}}, E_{\mathcal{AB}} \cup M)$ contains no green-white alternating cycles and no green-black alternating cycles.*

Note that if M is a valid partial matching for $J_{\mathcal{AB}}$ with $|M| = |V^J_{\mathcal{AB}}|/2 - 2$, then adding the last pair of vertices to M is a solution to the edge matching problem. The following theorem states that it is always possible to grow a valid partial matching into a valid partial matching, or into a solution to the edge matching problem.

Theorem 1. *Let $J_{\mathcal{AB}}$ be an edge matching graph and let M be a valid partial matching of $J_{\mathcal{AB}}$, with $|M| < |V^J_{\mathcal{AB}}|/2$. Then, there exists a pair of vertices $\{a, b\} \subseteq V^J_{\mathcal{AB}}$ not already in M and such that $M \cup \{\{a, b\}\}$ is a valid partial matching or a solution to the matching problem.*

Proof. See Appendix. □

The consequence of Theorem 1 is that one can find a solution to the edge matching problem by iteratively adding green edges one by one, each time ensuring that the new edge does not close an alternating cycle. One will never reach a point where no valid edge can be added, until a complete matching is obtained. Thus, Algorithm 1 is guaranteed to return a solution to the edge matching problem. In fact, the algorithm can be used to generate *all* optimal solutions, simply by exploring all choices on line 4. Moreover, the running time to find one solution is linear in the number of initial markers, by implementing the algorithm with hash tables that maintain black and white paths, which allows to choose a valid green edge and update structures in constant time within the loop.

[3] A white edge and a black edge have been added between the starting vertex and the ending vertex in the definition of the graph to prevent a green edge to link them prematurely into a path.

6 Putting Everything Together

Synthesizing all the previous remarks and propositions, we present the Block Ordering Algorithm (BOA), which is an algorithm that can output one optimal solutions to the BOP[4] in linear time[5].

Algorithm 1. EdgeMatchingAlgorithm

Require: A matching graph $J = (V^J, E^J)$.
Ensure: $Matching$, a set of green edges that constitutes a solution to the edge matching problem for J.
1: $Matching \leftarrow \{\}$ ▷ Represents the set of green edges
2: $Unused \leftarrow V^J$ ▷ Represents the set of vertices not already matched
3: **while** $|Matching| < |V^J|/2$ **do**
4: Choose $a, b \in Unused$ such that $Matching \cup \{\{a, b\}\}$ is valid
5: $Matching \leftarrow Matching \cup \{\{a, b\}\}$
6: $Unused \leftarrow Unused \setminus \{a, b\}$
7: **end while**
8: $Matching \leftarrow Matching \cup \{Unused\}$ ▷ Only one pair left in $Unused$ so add it

Algorithm 2. BOA: BlockOrderingAlgorithm

Require: Two sets of blocks $\mathcal{A}$ and $\mathcal{B}$
Ensure: Ordering of $\mathcal{A}$ called $P_\mathcal{A}$ and ordering of $\mathcal{B}$ called $P_\mathcal{B}$ such that the number of cycles of the breakpoint graph obtained from $P_\mathcal{A}$ and $P_\mathcal{B}$ is maximized.
1: $F_{\mathcal{AB}} \leftarrow$ Fragmented Breakpoint Graph for $\mathcal{A}$ and $\mathcal{B}$
2: $H_{\mathcal{AB}} \leftarrow$ Block Ordering Graph obtained from $F_{\mathcal{AB}}$
3: $J_{\mathcal{AB}} \leftarrow$ Edge Matching Graph from $H_{\mathcal{AB}}$
4: $(\Phi_\mathcal{A}, \Phi_\mathcal{B}) \leftarrow$ Blocks joined by β-one-sided edges & removed from $H_{\mathcal{AB}}$ ▷ Rem. 1
5: $(o_\mathcal{A}, o_\mathcal{B}) \leftarrow$ EdgeMatchingAlgorithm$(J_{\mathcal{AB}})$ ▷ Process all two-sided edges
6: **for** each white one-sided component X of $H_{\mathcal{AB}}$ **do** ▷ Process 1-sided components
7: $o \leftarrow$ some partial ordering induced by X
8: insert o anywhere between two blocks in $o_\mathcal{A}$ ▷ Proposition 5
9: **end for**
10: **for** each black one-sided component X of $H_{\mathcal{AB}}$ **do**
11: $o \leftarrow$ some partial ordering induced by X
12: insert o anywhere between two blocks in $o_\mathcal{B}$ ▷ Proposition 5
13: **end for**
14: Put blocks $(\Phi_\mathcal{A}, \Phi_\mathcal{B})$ back into $(o_\mathcal{A}, o_\mathcal{B})$ ▷ Revert action of line 4
15: $P_\mathcal{A} \leftarrow$ permutation induced by $o_\mathcal{A}$
16: $P_\mathcal{B} \leftarrow$ permutation induced by $o_\mathcal{B}$

Theorem 2 (Score of Block Ordering Algorithm outputs). *Let $(P_\mathcal{A}, P_\mathcal{B})$ be orderings of instance $(\mathcal{A}, \mathcal{B})$ as output by the Block Ordering Algorithm. Then,*

[4] Notice that there are non-deterministic steps in the algorithm: lines 5, 7, 8, 11, 12
[5] Each construction of an intermediate structure can be done in linear time in the number of initial markers (see section corresponding to each structure).

the score of $(P_{\mathcal{A}}, P_{\mathcal{B}})$ *is* $c(P_{\mathcal{A}}, P_{\mathcal{B}}) = \gamma + \frac{k}{2} + l_\alpha + l_\beta - \omega$*, where* γ *is the number of cycles in the fragmented breakpoint graph* $F_{\mathcal{AB}}$*,* k *is the number of two-sided edges,* l_α *is the number* α*-one-sided edges,* l_β *is the number of* β*-one-sided edges, and* ω *is the number of one-sided components in the BOG* $H_{\mathcal{AB}}$*.*

Proof. See Appendix. □

Theorem 3 (Optimality of the Block Ordering Algorithm). *The Block Ordering Algorithm outputs optimal orderings for every instance* $(\mathcal{A}, \mathcal{B})$*.*

Proof. See Appendix. □

7 Experiments

To illustrate the proposed algorithm, we have implemented it and used it to analyze both simulated and biological data. An interactive command-line version of the program, ordering simultaneously two block sets, is available at `www.mcb.mcgill.ca/~egaul/recomb2006`.

7.1 Biological Data Analysis

We first assess to what extent the solution to the BOP assembles the blocks in the correct order for an actual set of biological markers. The gene content of the X chromosomes of most mammals is generally well conserved, although several intra-chromosomal rearrangements differentiate them. Rearrangements in the X chromosome have been studied previously by Bourque *et al.*[1]. In their genome-wide mammalian gene order analysis, Ma *et al.* [2] have identified 90 unique markers located on the human and mouse chromosomes X. The order and orientation of these markers are quite different in the two species, and the reversal distance between them is 22. The advantage of using fully assembled genomes like human and mouse is that we know the correct gene arrangement.

We have artificially and randomly broken the corresponding signed permutations into blocks of various sizes, to mimic partially assembled genomes. We then ran the BOA algorithm on the two block sets and compared the predicted marker arrangement of the human chromosome to the actual arrangement by counting the number of breakpoints between the two. The breakpoint distance was normalized by $k-1$, where k is the number of blocks. This normalized error is thus the fraction of incorrect block joins.

Figure 5 shows how the relative error varies with the number blocks in the fragmented chromosome. The ideal score of 0 usually cannot be reached, because of the ambiguity that is part of the reconstruction process. The decrease in the relative error as the number of blocks increases from 2 to 9 is due to the fact that the absolute number of incorrect joins is relatively stable, while the normalizing factor increases.

To get an idea of the source of the ambiguity in the reconstruction and get some intuition about the impact of the topology of the connected components of

the block ordering graph, diverse values have been computed while executing the algorithm. The ambiguity is driven by four parameters: the number and size of one-sided components, and the number and size of two-sided components. The combinatorial choices in the edge matching algorithm suggests that the number of two-sided edges is critical in the accuracy of the reconstruction. Indeed, Fig. 5b and 5c clearly show that it is the number of two-sided components, and, in those, the number of two-sided edges, that are the chief causes of ambiguity, and thus of errors in the predictions. Block joins corresponding to β-one-sided edges are part of all optimal solutions, those corresponding to α-one-sided are part of most optimal solutions, and those corresponding to two-sided edges are generally part of a small number of optimal solutions.

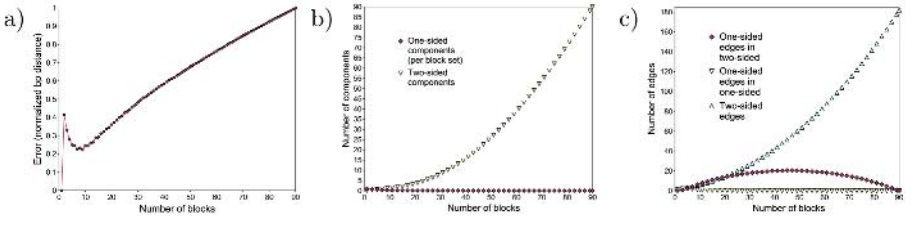


Fig. 5. (a) Error in reconstructing chromosomes X of human and mouse. Each chromosome has been randomly broken into a certain number of blocks (x-axis). The y-axis represents the mean of normalized breakpoint distances between the original and reconstructed genomes. (b) Number of connected components of each type in the BOG for the human-mouse chrX random fragmentation experiment. (c) Number of edges of each type in the BOG. For all three graphs, numbers plotted are the average over 1000 random fragmentations.

7.2 Simulated Data

To get an idea of the accuracy of the reconstruction as a function of the rearrangement distance between the two genomes being compared and as a function of the number of markers available, we used simulated data. Pairs of permutations of size 90 were generated to be separated by various reversal distances, and were then randomly fragmented into a certain number of blocks. After running BOA on this input, the normalized breakpoint error was computed. As seen in Fig. 6a, the results are as expected; it is harder, as the distance between the two original permutations or the number of blocks grows, to have good accuracy in the reconstruction. However, the error grows relatively slowly with the reversal distance, indicating that it might be possible to assemble pairs of genomes that are significantly more rearranged than human-mouse chrX. The third experiment has been designed to determine if it is possible to compensate for the lack of control on the number of blocks by adding more markers in the data. As Fig. 6b shows, the accuracy of the solution quickly improves with the number of markers. Indeed,

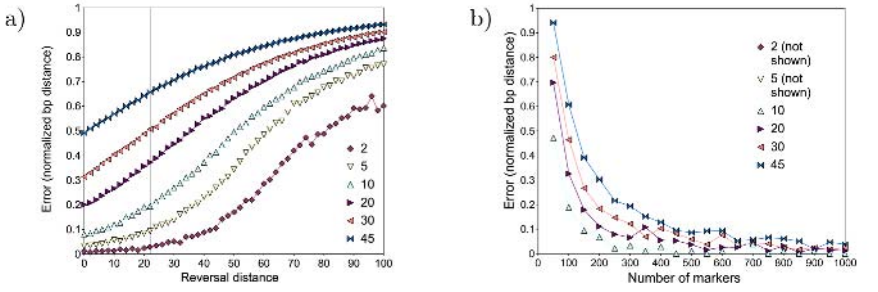


Fig. 6. (a) Error in reconstructing permutations of size 90, for pairs of permutations separated by various reversal distances. The vertical line at $x = 22$ represents the reversal distance between human and mouse chrX. Each curve corresponds to a different number of blocks in the fragmented genomes. (b) Error in reconstructing permutations at reversal distance 22, as a function of the number of markers available. Each curve corresponds to a different number of blocks.

as the number of markers increases, it becomes less likely that a break between block will land on a breakpoint between the two permutations.

8 Future Work and Conclusion

This work is a first step toward using the wealth of information coming from partially assembled genomes for studying genome rearrangements. A number of exciting algorithmic and evolutionary questions remain to be answered. We are currently working on generalizing the approach described here to optimize the actual reversal distance (rather than the number of cycles in the breakpoint graph). Generalizations to other rearrangement distances are also possible. Second, we plan to generalize our approach to handle multi-chromosomal genomes, although a new formalism will be needed to model chromosome ends. Finally, although we have shown in Sect. 7 that fairly accurate assemblies can be inferred even when the two genomes are fragmented in relatively small blocks, the accuracy could potentially be improved if more than two genomes were considered simultaneously. In a parsimony-based approach, one could consider a set of completely or partially assembled genomes labeling the leaves of a given phylogenetic tree, and ask to simultaneously order the partially assembled genomes and infer ancestral gene orders, in order to optimize some rearrangement-based criterion. Although this problem would be NP-hard for most optimization criteria, efficient heuristics may yield improved predictions of ancestral gene arrangements.

9 Supplementary Material

Implementation and proofs are available at:
`http://www.mcb.mcgill.ca/~egaul/recomb2006`

Acknowledgments

We thank Jian Ma and Webb Miller for the gene order data they provided. We thank Anne Bergeron and Nadia El-Mabrouk for their helpful comments. This work has been supported financially by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds Québécois pour la Recherche sur la Nature et les Technologies (FQRNT) team grant.

References

1. Bourque, G., Pevzner, P., Tesler, G.: Reconstructing the genomic architecture of ancestral mammals: lessons from human, mouse, and rat genomes. Genome Research **14(4)** (2004) 507–16
2. Ma, J., Zhang, L., Suh, B., Raney, B.J., Kent, W.J., Blanchette, M., Haussler, D., Miller, W.: Reconstructing contiguous regions of an ancestral genome. Genome Research **In press** (2006)
3. Murphy, W.J., Larkin, D.M., van der Wind, A.E., Bourque, G., Tesler, G., Auvil, L., Beever, J.E., Chowdhary, B.P., Galibert, F., Gatzke, L., Hitte, C., Meyers, S.N., Milan, D., Ostrander, E.A., Pape, G., Parker, H.G., Raudsepp, T., Rogatcheva, M.B., Schook, L.B., Skow, L.C., Welge, M., Womack, J.E., O'brien, S.J., Pevzner, P.A., Lewin, H.A.: Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. Science **309**(5734) (2005) 613–617
4. El-Mabrouk, N., Nadeau, J., Sankoff, D.: Genome halving. In Farach-Colton, M., ed.: Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching. Number 1448, Piscataway, NJ, Springer-Verlag, Berlin (1998) 235–250
5. Murphy, W.J., Bourque, G., Tesler, G., Pevzner, P., O'Brien, S.J.: Reconstructing the genomic architecture of mammalian ancestors using multispecies comparative maps. Hum Genomics **1**(1) (2003) 30–40
6. Tang, J., Moret, B.M.E.: Scaling up accurate phylogenetic reconstruction from gene-order data. Bioinformatics **19 Suppl 1** (2003) 305–312
7. Tesler, G.: GRIMM: genome rearrangements web server. Bioinformatics **18**(3) (2002) 492–493
8. Bourque, G., Tesler, G., Pevzner, P.A.: The convergence of cytogenetics and rearrangement-based models for ancestral genome reconstruction. Genome Res **16**(3) (2006) 311–313
9. Froenicke, L., Calds, M.G., Graphodatsky, A., Mller, S., Lyons, L.A., Robinson, T.J., Volleth, M., Yang, F., Wienberg, J.: Are molecular cytogenetics and bioinformatics suggesting diverging models of ancestral mammalian genomes? Genome Res **16**(3) (2006) 306–310
10. Peng, Q., Pevzner, P.A., Tesler, G.: The Fragile Breakage versus Random Breakage Models of Chromosome Evolution. PLoS Comput Biol **2**(2) (2006) e14
11. Sankoff, D., Trinh, P.: Chromosomal breakpoint reuse in genome sequence rearrangement. J Comput Biol **12**(6) (2005) 812–821
12. Margulies, E.H., Vinson, J.P., Miller, W., Jaffe, D.B., Lindblad-Toh, K., Chang, J.L., Green, E.D., Lander, E.S., Mullikin, J.C., Clamp, M., Program, N.I.S.C.C.S.: An initial strategy for the systematic identification of functional elements in the human genome by low-redundancy comparative sequencing. Proc Natl Acad Sci U S A **102**(13) (2005) 4795–4800

13. Siepel, A., Bejerano, G., Pedersen, J.S., Hinrichs, A.S., Hou, M., Rosenbloom, K., Clawson, H., Spieth, J., Hillier, L.W., Richards, S., Weinstock, G.M., Wilson, R.K., Gibbs, R.A., Kent, W.J., Miller, W., Haussler, D.: Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. Genome Res **15**(8) (2005) 1034–1050
14. Bourque, G., Pevzner, P.A.: Genome-Scale Evolution: Reconstructing Gene Orders in the Ancestral Species. Genome Res. **12**(1) (2002) 26–36
15. Bergeron, A.: A very elementary presentation of the Hannenhalli-Pevzner theory. Discrete Applied Mathematics **146**(2) (2005) 134–145
16. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals. Proceedings of the 27th Annual ACM Symposium on the Theory of Computing (1995) 178–187
17. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16) (2005) 3340–3346
18. Zheng, C., Lenert, A., Sankoff, D.: Reversal distance for partially ordered genomes. Bioinformatics **21**(suppl1) (2005) i502–508
19. Bergeron, A., Mixtacki, J., Stoye, J.: 10. In: Mathematics of Evolution and Phylogeny: the Inversion Distance Problem. First edn. Oxford University Press (2005) 262–290
20. Pevzner, P., Tesler, G.: Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. Proc Natl Acad Sci U S A **100**(13) (2003) 7672–7677

Evolution of Tandemly Repeated Sequences Through Duplication and Inversion

Denis Bertrand[1], Mathieu Lajoie[2,⋆], Nadia El-Mabrouk[2], and Olivier Gascuel[1]

[1] LIRMM, UMR 5506, CNRS et Univ. Montpellier 2, 161 rue Ada, 34392 Montpellier Cedex 5 France
{dbertran, gascuel}@lirmm.fr
[2] DIRO, Université de Montréal, CP 6128 succ Centre-Ville, Montréal QC, H3C 3J7, Canada
{lajoimat, mabrouk}@iro.umontreal.ca

Abstract. Given a phylogenetic tree T for a family of tandemly repeated genes and their *signed* order O on the chromosome, we aim to find the minimum number of inversions compatible with an evolutionary history of this family. This is the first attempt to account for inversions in an evolutionary model of tandemly repeated genes. We present a time-efficient branch-and-bound algorithm and show, using simulated data, that it can be used to detect "wrong" phylogenies among a set of putative ones for a given gene family. An application on a published phylogeny of KRAB zinc finger genes is presented.

Keywords: gene family, gene order, inversion, duplication, phylogeny.

1 Introduction

A large fraction of most genomes consists of repetitive DNA sequences. In mammals, up to 60% of the DNA is repetitive. A large proportion of such repetitive sequences is organized in tandem: copies of a same basic unit that are adjacent on the chromosome. The duplicated units can be small (from 10 to 200 bps) as it is the case of micro- and minisatellites, or very large (from 1 to 300 kb) and potentially contain several genes. Such large segment duplication is a primary mechanism for generating gene clusters on chromosomes.

Many gene families of the human genome are organized in tandem, including HOX genes [31], immunoglobulin and T-cell receptor genes [21], MHC genes [20] and olfactory receptor genes [11]. Reconstructing the duplication history of each gene family is important to understand the functional specificity of each copy, and to provide new insights into the mechanisms and determinants of gene duplication, often recognized as major generators of novelty at the genome level.

Based on the initial evolutionary model of tandemly repeated sequences introduced by Fitch [9], a number of recent studies have considered the problem of reconstructing a tandem duplication history of a gene family [5,6,7,16,28,32].

⋆ The two first authors have contributed equally to this work.

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 129–140, 2006.

These are essentially phylogenetic inference methods using the additional constraint that the resulting tree should induce a duplication history concordant with the given gene order. When a phylogeny is already available, a linear-time algorithm can be used to check whether it is a duplication tree [32]. However, even for gene families that have evolved through tandem duplications, it is often impossible to reconstruct a duplication history [7]. This can be explained by the fact that the duplication model is oversimplified, and other evolutionary events have occurred, such as gene losses or genomic rearrangements.

Evidence of gene inversion is observed in many tandemly repeated gene families, such as zinc finger (ZNF) genes, where gene copies have different transcriptional orientations [25]. Although genome rearrangement with inversions has received large attention in the last decade [14, 17, 4, 26, 2], beginning with the polynomial-time algorithm of Hannenhalli and Pevzner for computing the reversal distance between two signed gene orders [14], inversions have never been considered in the context of reconstructing a duplication history from a gene tree. In the case of general segmental duplications (not necessarily in tandem), potential gene losses have been considered to explain the non congruence between a gene tree and a species tree [12, 19, 18, 3]. Similarly, in the case of tandem duplication, the non-congruence between a gene tree and an observed gene order can be naturally explained by introducing the possibility of segmental inversions.

In this paper, our goal is to infer an evolutionary history of a gene family accounting for both tandem duplications and inversions. As the number of such possible evolutionary histories may be very large, we restrict ourselves to finding the minimum number of inversions required to explain a given ordered phylogeny. As a first attempt, we only considered tandem duplications involving single genes. Though the model described by Fitch [9] allows for simultaneous duplications of several gene copies, single duplications are known to be predominant over multiple duplications [1, 9, 28].

After describing the evolutionary models in section 2 and the optimization problem in section 3, we present our main branch-and-bound algorithm in section 4. Finally, in section 5, we test the algorithm's time-efficiency on simulated data and show its usefulness to detect, among a set of possible phylogenies, the "wrong" ones. An application on KRAB zinc finger genes is presented.

2 The Evolutionary Model

2.1 Duplication Model

This model, first introduced by Fitch [9], is based on unequal recombination during meiosis, which is assumed to be the sole evolutionary mechanism (except point mutations) acting on sequences. Consequently, from a single sequence, the locus grows through a series of consecutive duplications, giving rise to a sequence of n adjacent copies of homologous genes *having the same transcriptional orientation*. We denote by $O = (l_1, \cdots l_n)$ the observed ordered sequence of extant gene copies.

A *tandem duplication history* (or just *duplication history* for brevity) is the sequence of tandem duplications that have generated O. It can be represented by a rooted tree with n ordered leaves corresponding to the n ordered genes, in which internal nodes correspond to duplication events (Figure 1.a). Duplications may be *simple* (duplication of a single gene) or multiple (simultaneous duplication of neighboring genes). In this paper, we only consider simple duplications.

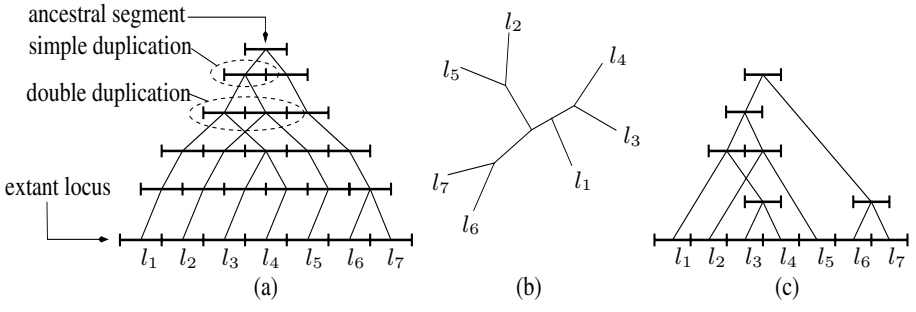


Fig. 1. (a) Duplication history; each segment represents a copy. (b) The unrooted duplication tree corresponding to history (a). (c) The duplication tree corresponding to history (a).

In a real duplication history, the time intervals between consecutive duplications are known, and the internal nodes are ordered from top to bottom according to the moment they occurred in the course of evolution. However, in the absence of a molecular clock mode of evolution, it is impossible to recover the order of duplication events. All we can infer from gene sequences is a phylogeny with ordered leaves (Figure 1.b). Formally, an *ordered phylogeny* is a pair (T, O) where T is a phylogeny and O is the ordered sequence of its leaves. According to this model, all the genes have the same transcriptional orientation.

If an ordered phylogeny (T, O) can be explained by a duplication history $\mathcal{H}$, we say that (T, O) is *compatible* with $\mathcal{H}$, and that $\mathcal{H}$ *is a duplication history of* (T, O). If (T, O) is compatible with at least one duplication history, it is called a *duplication tree*. Choosing appropriate roots for unrooted duplication trees is discussed in [10] (Figure 1.c).

In the rest of this paper, a *duplication tree* will refer to a *simple rooted duplication tree*, that is a rooted duplication tree that is compatible with at least one history involving only simple duplications. Unless otherwise stated, all the phylogenies are rooted.

2.2 A Duplication/Inversion Model

Many tandemly repeated gene families contain members in both transcriptional orientations. The simple duplication model is thus inadequate to describe their evolution. To circumvent this limitation, we propose an extended model of duplication which includes inversions. Thereafter, the transcriptional orientations

of the genes in a *signed* ordered phylogeny (T, O) are specified by signs $(+/-)$ in O. Thus O is formally a signed permutation of the leaves of T. We denote by $d_{inv}(O_i, O_j)$ the inversion distance between the two signed permutations O_i and O_j. Note that a signed ordered phylogeny (T, O) cannot be a duplication tree unless all the genes in O have the same sign (although this is not a sufficient condition).

Definition 1. *A* simple duplication/inversion history *(or just* dup/inv history*) of length k is an ordered sequence $\mathcal{H}_k = ((T_1, O_1), ..., (T_{k-1}, O_{k-1}), (T_k, O_k))$ where :*

1. *Every (T_i, O_i) is a* signed *ordered phylogeny.*
2. *$T_1 = v$ is a single leaf phylogeny and $O_1 = (\pm v)$ one of the two trivial orders.*
3. *For $0 < i < k$,*
   - *if $T_{i+1} = T_i$, then $d_{inv}(O_i, O_{i+1}) = 1$. This corresponds to one inversion event.*
   - *if $T_{i+1} \neq T_i$, then T_{i+1} is obtained from T_i by adding two children u and w to one of its leaf v. In this case O_{i+1} is obtained from O_i by replacing $\pm v$ by $(\pm u, \pm w)$. This corresponds to a simple duplication event.*

3 An Inference Problem

A signed ordered phylogeny is not necessarily compatible with a duplication history. The following lemma shows that additional inversions can always be used to infer a possible evolutionary history for the gene family.

Lemma 1. *A signed ordered phylogeny (T, O) is compatible with at least one simple duplication/inversion history.*

Proof. According to Definition 1, obtain a duplication tree (T, O') by successive duplication events. Then, transform O' into O by applying the required inversions. □

As the number of possible dup/inv histories explaining (T, O) can be very large, we restrict ourselves to finding the minimum number of events involved in such evolutionary histories. More precisely, as the number of simple duplications is fixed by T, we are interested in finding the minimum number of inversions involved in a dup/inv history. The next theorem shows that if i is the minimum number of inversions needed to transform O into O' such that (T, O') is a duplication tree, any dup/inv history of (T, O) contains at least i inversions.

Theorem 1. *Let (T, O) be a signed ordered phylogeny. For any dup/inv history $\mathcal{H}$ with i inversions leading to (T, O), there exists a duplication tree (T, O') such that $d_{inv}(O, O') \leq i$.*

Proof by induction.

- Base case: Let $\mathcal{H}_1 = (T_1, O_1)$ be a dup/inv history with no duplication or inversion. Clearly $(T, O') = (T_1, O_1)$ is a duplication tree.
- Induction step (on the number k of events):
  Let $\mathcal{H}_{k+1} = ((T_1, O_1), ..., (T_k, O_k), (T_{k+1}, O_{k+1}))$ be a dup/inv history involving $k+1$ events and i inversions and $\mathcal{H}_k = ((T_1, O_1), ..., (T_k, O_k))$. From Definition 1, there are two possibilities:
  - If $T_{k+1} = T_k$, then the last event is an inversion, and $\mathcal{H}_k$ is a dup/inv history involving $i-1$ inversions. By induction hypothesis, there exists a duplication tree (T_k, O'_k) such that $d_{inv}(O_k, O'_k) \leq i-1$. Let O_{k+1} be the order obtained from O_k by applying the last inversion. Then we have $d_{inv}(O_{k+1}, O'_k) \leq d_{inv}(O_k, O'_k) + 1 \leq i$.
  - If $T_{k+1} \neq T_k$, the last event is a duplication, that is a leaf $\pm v$ of (T_k, O_k) is replaced by two consecutive leaves $(\pm u, \pm w)$ in (T_{k+1}, O_{k+1}). Let (T_k, O'_k) be the duplication tree associated to $\mathcal{H}_k$ and suppose that all elements of O'_k are positive. If we have $+v$ in O_k, we obtain O'_{k+1} by replacing $+v$ by $(+u, +w)$ in O'_k. Otherwise we have $-v$ in O_k and we obtain O'_{k+1} by replacing $+v$ by $(+w, +u)$ in O'_k. Thus, $d_{inv}(O_{k+1}, O'_{k+1}) = d_{inv}(O_k, O'_k) \leq i$ and (T_{k+1}, O'_{k+1}) is a duplication tree. The case where the elements of O'_k have a negative sign is similar. □

Corollary 1. *Let (T, O) be a signed ordered phylogeny and (T, O') a duplication tree such that $d_{inv}(O, O') = i$ is minimum. There exists a dup/inv history $\mathcal{H}$ for (T, O) with exactly i inversions, which is optimal.*

Proof. The existence of $\mathcal{H}$ for (T, O) with exactly i inversions follows directly from the proof of Lemma 1. The number i of inversions in $\mathcal{H}$ must be optimal, otherwise, from Theorem 1, it would contradicts the hypothesis that $d_{inv}(O, O') = i$ is minimum. □

Corollary 1 allows to reformulate our problem in the following way :

Minimum-Inversion Duplication problem
Input: A signed ordered phylogeny (T, O),
Output: An order O' such that (T, O') is a duplication tree and $d_{inv}(O, O')$ is minimal.

4 A Branch-and-Bound Algorithm

We begin by briefly summarizing the Hannenhalli-Pevzner method [14], as it will be used in our approach.

4.1 Hannenhalli-Pevzner (HP) Algorithm

Given two signed permutations O, O' of size n on the same set of genes, the problem is to find the minimal number $d_{inv}(O, O')$ of inversions required to transform O to O' (or similarly O' to O). The algorithm is based on a bicolored

graph, called the *breakpoint graph*, constructed from the two permutations as follows: if gene x of O has a positive sign, replace it by the pair $x_t x_h$, and if it is negative, by $x_h x_t$. Then the vertices of the graph are just the x_t and the x_h for all genes x. The graph contains two classes of edges: the real and desired edges (as named in [24]). Any two vertices which are adjacent in O, other than x_t and x_h deriving from the same x, are connected by a *real edge*, and any two adjacent in O', by a *desired edge*. This graph decomposes naturally into a set of c disjoint color-alternating cycles. An important property of the graph is its decomposition into components, where a *component* is a maximal set of "crossing" cycles.

Based on this graph, the inversion distance can be computed according to the following formula [14]:

$$d_{inv}(O, O') = n + 1 - c + h + f,$$

where h and f are quantities related to the presence of "hurdles" (components of a particular type). As the probability for a component to be a hurdle is low, h and f are usually close to 0. Therefore, the number of cycles c is the dominant parameter in the formula. In other words, the more cycles there are, the less inversions we need to transform O into O'.

4.2 Enumerating the Compatible Orders

We say that an order O' is *compatible* with a phylogeny T iff (T, O') is a duplication tree. To enumerate all the orders compatible with T, we associate a binary variable b_i to each internal node i of T. Each b_i defines an order relation between the left and right descendant leaves of i. By setting b_i to 0, we make all the left descendants smaller than the right ones. Conversely, by setting b_i to 1, all left descendants are considered larger than the right ones (see Figure 2.a.b). Assigning a value to all internal nodes of T defines a total order O' on its leaves: the order between two leaves is determined by the b_i value of their closest common ancestor. Otherwise, the order is partial since some pairs of leaves are incomparable. We will denote such a partial order as O^*. Note that every such order admits two transcriptional orientations according to our definition of a duplication tree.

Lemma 2. *An order O' is compatible with T iff it is defined by an assignment of all the binary variables b_i in T and all the genes have the same sign.*

Therefore, if n is the number of leaves in T, there are 2^{n-1} possible assignments of the b_i variables, each with two possible transcriptional orientations. This leads to 2^n distinct orders O' compatible with T. Hereafter, for clarity of presentation, we will only consider one of the two orientations.

4.3 A Lower Bound for the Inversion Distance

To avoid computing $d_{inv}(O, O')$ for each of the 2^n orders O' compatible with T, we consider a branch-and-bound strategy similar to the one used in [33]. The idea is to compute a lower bound on $d_{inv}(O, O')$ as we progressively define O^*

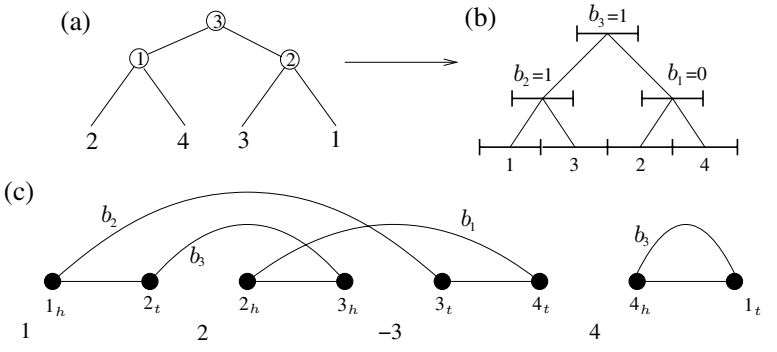


Fig. 2. (a) A phylogeny with an appropriate depth-first labeling of the internal nodes; (b) The duplication tree corresponding to an assignment of the b_i variables of (a); (c) The breakpoint graph illustrating the difference between the gene order $O' = (1, 3, 2, 4)$ obtained from the duplication tree (b) and the gene order $O = (1, 2, -3, 4)$ observed in the genome. *Desired edges* (curved edges) are added in the same order as the corresponding b_i values (b_1 then b_2 then b_3). For simplicity, the genome is assumed to be circular (gene 1 next to gene 4).

by updating the breakpoint graph of (O, O^*). In order to progressively construct this graph, it is essential to define the b_i values in a depth-first manner according to T: the binary variables of all the descendant nodes of i should be defined before b_i. This insures that the two subtrees of i have a total order on their leaves.

Consequently, if we set b_i to 0, the greatest left descendant leaf l_{max} of node i will immediately precedes its smallest right descendant leaf r_{min} in O'. Conversely, if b_i is set to 1, the greatest right descendant r_{max} will immediately precede the smallest left descendant l_{min}. Therefore, the assignment of a b_i value allows us to add a desired edge in the breakpoint graph between l_{max} and r_{min} (or r_{max} and l_{min}) (see Figure 2.c).

Let O^* be the partial order obtained at a given stage of the procedure. Let e be the number of cycles and p the number of remaining edges to place in the corresponding incomplete breakpoint graph. Each of the remaining edges can create at most one cycle, ending with a breakpoint graph with at most $c = e + p$ cycles. Thus, any total order O' that can be obtained from the partial order O^* is such that:

$$d_{inv}(O, O') = n + 1 - c + h + f \geq n + 1 - c \geq n + 1 - p - e = d^*_{inv}(O, O^*).$$

The branch-and-bound algorithm proceeds as follows. An initial assignment of all binary variables is considered and the corresponding exact reversal distance $d_{inv}(O, O')$ is computed using the HP algorithm. Each following step re-assigns the binary variables in a depth-first manner. If the partial order O^* obtained is such that $d^*_{inv}(O, O^*) \geq \min_{inv}$, where $\min_{inv}$ is the lowest inversion distance obtained from the previous steps, we backtrack to the last node (closest to the root) that has not been re-assigned twice. This is justified by the fact that any

total order that can be obtained from O^* cannot be smaller than the current best value. Every time we reach a leaf, we use the HP algorithm to compute $d_{inv}(O, O')$ and update $\min_{inv}$.

5 Results

5.1 Branch-and-Bound Efficiency

To test the efficiency of the branch-and-bound algorithm, we generated 3 sets of 500 phylogenies each with respectively 10, 20 and 40 leaves using r8s [23]. We then defined arbitrarily compatible orders to obtain a total of 1,500 duplication trees. For each of them, we performed 1, 2, 4 and 8 inversions to obtain 12 datasets containing a total of 6,000 signed ordered phylogenies which are no longer duplication trees.

We applied our algorithm on each dataset and measured the execution time (on a Pentium 4) and the average fraction of nodes explored in the search space. Results are given in Table 1. We observe that the algorithm is very efficient and can be used on relatively important phylogenies within reasonable time.

Table 1. Average fraction of nodes explored in the search tree during the branch-and-bound / Execution time (in seconds) for the 500 signed ordered phylogenies

	1 inversion	2 inversions	4 inversions	8 inversions
10 leaves	1×10^{-2} / 13	2×10^{-2} / 20	6×10^{-2} / 35	0.1/51
20 leaves	3×10^{-5} / 15	8×10^{-5} / 20	2×10^{-4} / 37	2×10^{-3} / 90
40 leaves	7×10^{-11} / 17	2×10^{-10} / 24	1×10^{-9} / 39	2×10^{-8} / 112

5.2 Application on Simulated Data

We applied our algorithm on simulated data to verify how it could be used to validate inferred phylogenies on tandemly repeated gene families. Using the simulation protocol described in the previous section, we randomly generated 500 duplication trees with 15 leaves. For each one of them we performed 0, 2, 4 and 6 inversions to obtain 4 datasets containing a total of 2,000 signed ordered phylogenies. These are the observable states (T_{true}, O) resulting from "true" duplication/inversion histories. For each T_{true}, we then randomly generated two "wrong" (but close) phylogenies T_{wrong}, that can be obtained by applying respectively one or two Nearest Neighbor Interchange rearrangements (NNI) [27]. Those "wrong" phylogenies can be seen as the ones we could obtain from biological data when a few nodes have weak statistical support. Finally, we used our algorithm to compute the minimum number of inversions inv() necessary in a simple duplication/inversion history to explain each (T_{true}, O) and all its corresponding T_{wrong}. The averaged results are presented in Table 2.

Results can be interpreted as follows. For a wrong phylogeny T_{wrong}, 50% of the time on average our algorithm reports an excess of inversions, otherwise it

Table 2. Percentage of times $\text{inv}(T_{wrong}, O)$ is less, equal or greater than $\text{inv}(T_{true}, O)$. Averaged over all possible neighbors for each of the 500 phylogenies.

	Trees distant from one NNI				Trees distant from two NNI			
$\text{inv}(T_{wrong}, O)$	0 *inv.*	2 *inv.*	4 *inv.*	6 *inv.*	0 *inv.*	2 *inv.*	4 *inv.*	6 *inv.*
$< \text{inv}(T_{true}, O)$	0.0	0.2	0.8	1.6	0.0	0.2	0.08	1.6
$= \text{inv}(T_{true}, O)$	49.0	50.4	57.6	68.0	36.6	35.0	41.2	54.4
$> \text{inv}(T_{true}, O)$	51.0	49.4	41.6	30.4	66.4	64.8	58.0	44.0

reports the same number of inversions compared to the true phylogeny T_{true}. Suppose that we are presented some ordered phylogenies. One is correct and the others differ by a few NNIs. According to Table 2, for wrong trees, the algorithm almost always reports the same number of inversions or more as in the true tree. Thus, choosing the phylogeny with the lowest number of inversions is either a winning strategy (roughly 50% of the time) or useless, but is almost never misleading. Of course, this ability to discard wrong phylogenies decreases as the true number of inversions increases.

5.3 Application on Biological Data

The KRAB-zinc finger gene family encodes for transcription factors. It contains more than 400 active members physically grouped into clusters. In a recent study [13], Hamilton *et al.* proposed a phylogeny of the primate specific ZNF91 sub-family based on their tether[1] and flanking sequences. This phylogeny (obtained by Neighbor-Joining [22]) contains a monophyletic group of 6 genes clustered at the telomere of HSA4p, which may have been derived from a single ancestor through successive tandem duplications.

We applied our algorithm on this cluster using the proposed phylogeny, and found that a duplication/inversion history would require at least 4 inversions, which seems relatively high considering that only 6 genes are involved.

To test whether a "better" phylogeny could be proposed, we used the MrBayes software [8] to obtain a sample from the posterior probability distribution of all possible phylogenies. The tether (+100 flanking bp) sequences were downloaded from the Human KZNF Gene Catalog[2] [15] and aligned using ClustalW [29] with default settings. The ZNF160 tether sequence was used as an outgroup to obtain a rooted tree. We performed 500,000 MCMC generations with MrBayes under the GTR model [30] and a gamma-shaped rate variation with a proportion of invariable sites. Convergence was easily attained and the experiment was repeated three times with similar results. Finally we applied our algorithm on the sampled phylogenies and observed that the best one (p=0.4) is compatible with an optimal duplication/inversion history involving only two inversions. Phylogenies are presented in Figure 3.

[1] The region upstream from the first finger.

[2] `http://znf.llnl.gov/catalog/`

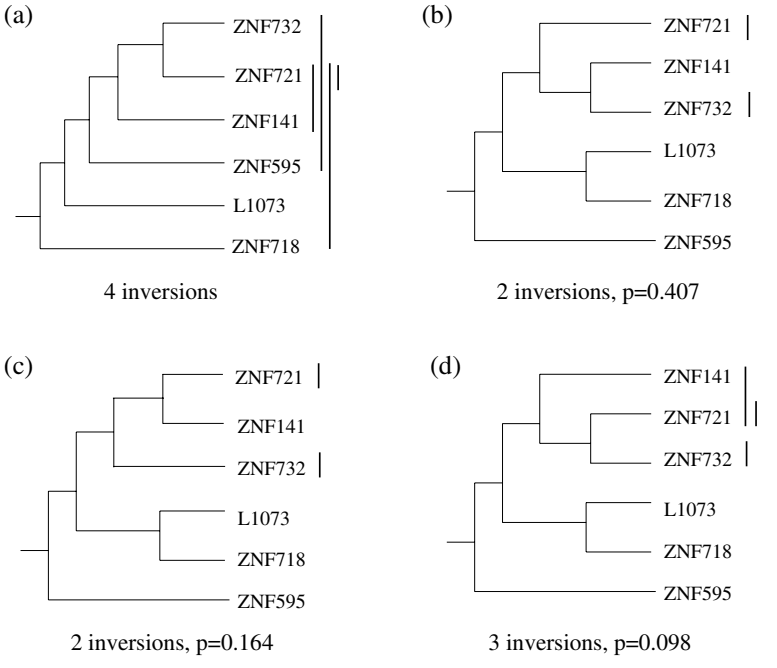


Fig. 3. Different phylogenies for the ZNF141 clade on human chromosome 4, with the associated minimal number of inversions in a dup/inv history. The black vertical lines represent an optimal sequence of inversions leading to the *signed* gene order observed on the chromosome: (+ZNF595,+ZNF718,+L1073,−ZNF732,+ZNF141,−ZNF721). (a) The phylogeny published in [13] requires 4 inversions, which is relatively high for 6 genes; (b,c,d) The 3 best phylogenies we obtained with MrBayes, and their associated probabilities. The first two ones require only 2 inversions, which is optimal for this order. The position of the root was determined using ZNF160 as an outgroup.

6 Conclusion

This work represents the first attempt to account for inversions in an evolutionary model of tandemly repeated genes. We presented a time-efficient branch-and-bound algorithm for finding the minimal number of inversions in an evolutionary history of a gene family characterized by an ordered phylogeny. Though only simple duplications were considered here, the model has been shown useful to select an appropriate phylogeny among a set of possible ones. These are encouraging results that motivate further extensions.

The next step of this work will be to account for multiple duplications in the evolutionary model. Another important generalization will be to consider a family of tandemly duplicated genes with orthologs in two or more genomes. For example, Shannon *et al.* [25] identified homologous ZNF gene family regions in human and mouse. A phylogenetic tree involving such tandemly repeated genes in human and mouse clusters was established. It would be of major interest to

develop an algorithm allowing to explain such a phylogeny based on an evolutionary model involving tandem duplication, inversion and speciation events.

Acknowledgments. The authors wish to thank M. Aubry and H. Tadepally for their help on zinc finger genes. This work was supported by grants from the Natural Sciences and Engineering Research Council of Canada (N.E.M.) and the Canadian Institutes of Health Research (M.L.).

References

1. G. Benson and L. Dong. Reconstructing the duplication history of a tandem repeat. In *Proceedings of Intelligent Systems in Molecular Biology (ISMB1999), Heidelberg, Germany*, pages 44–53. AAAI, 1999.
2. A. Bergeron, J. Mixtacki, and J. Stoye. Reversal distance without hurdles and fortresses. volume 3109 of *LNCS*, pages 388 - 399. Springer-Verlag, 2004.
3. K. Chen, D. Durand, and M. Farach-Colton. Notung: Dating gene duplications using gene family trees. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (RECOMB 2000)*, New York, 2000. ACM.
4. N. El-Mabrouk. Genome rearrangement by reversals and insertions/deletions of contiguous segments. In *CPM 2000*, volume 1848 of *LNCS*, pages 222- 234, 2000.
5. O. Elemento and O. Gascuel. A fast and accurate distance-based algorithm to reconstruct tandem duplication trees. *Bioinformatics*, 18:92–99, 2002.
6. O. Elemento and O. Gascuel. An exact and polynomial distance-based algorithm to reconstruct single copy tandem duplication trees. *Journal of Discrete Algorithms*, 2(2-4):362–374, 2005.
7. O. Elemento, O. Gascuel, and M-P. Lefranc. Reconstructing the duplication history of tandemly repeated genes. *Molecular Biology and Evolution*, 19:278–288, 2002.
8. J.P. Huelsenbeck F. Ronquist. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–4, 2003.
9. W.M. Fitch. Phylogenies constrained by cross-over process as illustrated by human hemoglobins in a thirteen-cycle, eleven amino-acid repeat in human apolipoprotein A-I. *Genetics*, 86:623–644, 1977.
10. O. Gascuel, D. Bertrand, and O. Elemento. Reconstructing the duplication history of tandemly repeated sequences. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny*, pages 205–235. Oxford University Press, 2005.
11. G. Glusman, I. Yanai, I. Rubin, and D. Lancet. The complete human olfactory subgenome. *Genome Research*, 11(5):685–702, 2001.
12. R. Guigó, I. Muchnik, and T.F. Smith. Reconstruction of ancient molecular phylogeny. *Molecular Phylogenetics and Evolution*, 6:189–213, 1996.
13. A.T. Hamilton, S. Huntley, M. Tran-Gyamfi, D.M. Baggott, L.Gordon, and L. Stubbs. Evolutionary expansion and divergence in the znf91 subfamily of primate-specific zinc finger genes. *Genome Research*, 16(5):584–594, 2006.
14. S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *J. ACM*, 48:1–27, 1999.
15. S. Huntley, D.M. Baggot, A.T. Hamilton, S. Yang M. TranGyamfi, J. Kim, L. Gordon, E. Branscomb, and L. Stubbs. A comprehensive catalogue of human krab-associated zinc finger genes: Insights into the evolutionary history of a large family of transcriptional repressors. *Genome Research*, 16:669–677, 2006.

16. D. Jaitly, P. Kearney, G. Lin, and B. Ma. Methods for reconstructing the history of tandem repeats and their application to the human genome. *Journal of Computer and System Sciences*, 65:494–507, 2002.
17. H. Kaplan, R. Shamir, and R. E. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing*, 29:880–892, 2000.
18. Bin Ma, Ming Li, and L. Zhang. On reconstructing species trees from gene trees in term of duplications and losses. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Biology (RECOMB 98)*, pages 182–191, New York, 1998. ACM.
19. R.D.M Page and M.A. Charleston. Reconciled trees and incongruent gene and species trees. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 37:57–70, 1997.
20. J. Robinson, M.J. Waller, P. Parham, N. de Groot, R. Bontrop, L.J. Kennedy, P. Stoehr, and S.G. Marsh. IMGT/HLA and IMGT/MHC: sequence databases for the study of the major histocompatibility complex. *Nucleic Acids Research*, 31(1):311–4, 2003.
21. M. Ruiz, V. Giudicelli, C. Ginestoux, P. Stoehr, J. Robinson, J. Bodmer, S.G. Marsh, R. Bontrop, M. Lemaitre, G. Lefranc, D. Chaume, and M-P. Lefranc. IMGT, the international ImMunoGeneTics database. *Nucleic Acids Research*, 28:219–221, 2000.
22. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.
23. M.J. Sanderson. r8s; inferring absolute rates of evolution and divergence times in the absence of a molecular clock. *Bioinformatics*, 19:301 - 302, 2003.
24. J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*, chapter 7. PWS Pub. Co., 1997. SET j 97:1 1.Ex.
25. M. Shannon, A.T. Hamilton, L. Gordon, E. Branscomb, and L. Stubbs. Differential expension of Zinc- Finger transcription factor loci in homologous human and mouse gene clusters. *Genome Research*, 13:1097 - 1110, 2003.
26. A. Siepel. Algorithm to find all sorting reversals. In *Proceedings of the second conference on computational molecular biology (RECOMB'02)*, pages 281 - 290. ACM Press, 2002.
27. D.L. Swofford, P.J. Olsen, P.J. Waddell, and D.M. Hillis. *Molecular Systematics*, chapter Phylogenetic Inference, pages 407–514. Sinauer Associates, Sunderland, Massachusetts, 1996.
28. M. Tang, M.S. Waterman, and S. Yooseph. Zinc finger gene clusters and tandem gene duplication. In *Proceedings of International Conference on Research in Molecular Biology (RECOMB2001)*, pages 297–304, 2001.
29. J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673 - 4680, 1994.
30. P. Waddel and M. Steel. General time reversible distances with unequal rates across sites: Mixing t and inverse gaussian distributions with invariant sites. *Molecular Phylogeny and Evolution*, 8:398–314, 1997.
31. J. Zhang and M. Nei. Evolution of antennapedia-class homeobox genes. *Genetics*, 142(1):295–303, 1996.
32. L. Zhang, B. Ma, L. Wang, and Y. Xu. Greedy method for inferring tandem duplication history. *Bioinformatics*, 19:1497–1504, 2003.
33. C. Zheng, A. Lenert, and D. Sankoff. Reversal distance for partially ordered genomes. *Bioinformatics*, 21:i502 - i508, 2003.

A PQ Framework for Reconstructions of Common Ancestors and Phylogeny

Laxmi Parida

Computational Biology Center, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
parida@us.ibm.com

Abstract. Various international efforts are underway to catalog the genomic similarities and variations in the human population. Some key discoveries such as inversions and transpositions within the members of the species have also been made over the years. The task of constructing a phylogeny tree of the members of the same species, given this knowledge and data, is an important problem. In this context, a key observation is that the "distance" between two members, or member and ancestor, within the species is small. In this paper, we pose the tree reconstruction problem exploiting some of these peculiarities. The central idea of the paper is based on the notion of minimal consensus PQ tree T of sequences introduced in [29]. We use a modified PQ structure (termed oPQ) and show that both the number and size of each T is $\mathcal{O}(1)$. We further show that the tree reconstruction problem is statistically well-defined (Theorem 7) and give a simple scheme to construct the phylogeny tree and the common ancestors. Our preliminary experiments with simulated data look very promising.

Keywords: PQ tree, inversion, reversal, transposition, genome rearrangement, phylogeny, evolution, genealogy, common ancestor, tree construction.

1 Introduction

Various international efforts are underway to catalog the genomic similarities and variations in the human population [7, 5, 3, 36]. As the study progresses, data in the form of genomic markers is becoming available, with due respect to individuals' and groups' privacy, for public study and use. Combined with recent discoveries of inversion and transposition within the human species, this opens up the potential for using large scale rearrangements to reconstruct the genealogy tree of the human population.

We first give a brief summary of discovered inversions and transpositions within the human population and then briefly review the computational methods being used by the bioinformatics community to tackle the problem of reconstructing phylogeny trees.

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 141–155, 2006.

Inversions along a chromosome are frequently observed by comparing closely related species: for example, chimpanzee chromosome 19 and human chromosome 17 [13], mouse chromosome 16 and human chromosome 21 [18]. These are generally very long inversions that are observed as reversed gene orders [24]. Moreover, with the most recent builds of the chimpanzee genome, a total of 1,576 putative regions of inverted orientation, covering more than 154 mega-bases, of all sizes between the human and chimpanzee genomes have been observed [28]. However inversions have been seen across humans: X chromosome [33] and a 3 Mb inversion on the short arm of the Y chromosome [11]. Human inversions occur at a low but detectable frequency. The ones that are large enough to be detected by standard cytogenetic analysis occur at a frequency of 1-5 per 10,000 individuals [16]. The inversions across humans are of particular interest, since often the recombination in the inverted segments in heterozygotes lead to heritable disorders [17, 12].

Secondly, inversions also have a potential for explaining the geographic distribution of the human population: a reconstruction of the prehistoric human colonization of the planet [5,3,36]. The X-chromosome inversion is seen in populations of European descent at a frequency of about 18% [33]. Further large chromosomal segment inversion have been seen in humans: [19] reports a paracentric[1] inversion polymorphism spanning larger than 2.5 Mb segment in chromosome band $8p23.1-8p22$ and [14] reports a 900-Kb inversion on chromosome 17q21-31. The second inversion is seen at the rate of 20% in Europeans and almost absent in East Asians and rare in Africans. Also (benign) inversions at much higher rates are seen in chromosomes 2 and 9 and an inversion in chromosome 8 is seen more often in families of Mexican-American descent [21].

Large chromosomal rearrangement polymorphisms such as deletions or duplications are apparent by loss or gain of heterezygosity. However inversions are difficult to detect and may go unnoticed if the inverted segment is small.

The inversions may occur in coding, non-coding, or intra-gene regions of the chromosome. Hence a model that tracks the gene orders of the chromosome is inadequate for modeling segment inversions. Instead, these inversions [19,14] are being discovered and reported in terms of the order of the labeled STRP's (Short Tandem Repeat Polymorphisms). See Figure 1 for two instances of inversions in the human chromosomes. Further, unlike genes, these markers are not signed (say, as used in [4]). Also the ancestral segment is unknown, i.e. it is unclear which order of the segment came first.

Translocations [2] have also been observed in humans [9, 10], although these have been mostly of single genes and generally associated with a disorder. It is

[1] An inversion not involving the centromere.

[2] Translocation involves two nonhomologous chromosomes. Following a break and subsequent reunion in each of the chromosomes, a segment of one chromosome becomes attached to the other chromosome and vice versa. For example if chromosome 2 is defined as $g_1^2 g_2^2 g_3^2 g_4^2$ and chromosome 6 as $g_1^6 g_2^6 g_3^6 g_4^6 g_5^6$, after the translocation chromosome 2 is redefined as $g^1 g_2^2 g_4^6 g_5^6$ and chromosome 6 is redefined as $g_1^6 g_2^6 g_3^6 g_3^2 g_4^2$. This is an example of *balanced* translocation. On the other hand, *Robertsonian* translocation involves loss of genetic material.

believed that as we progressively learn about individual differences, more such variations, transpositions [3] or inversions, will surface. In fact according to [14], *these* (inversions) *may be only the tip of the iceberg.*

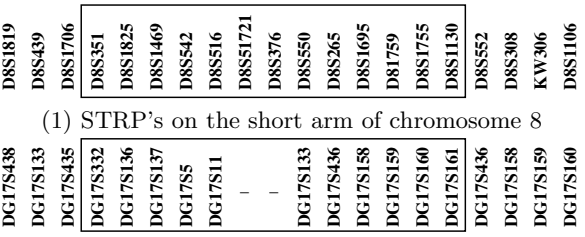


Fig. 1. The Short Tandem Repeat Polymorphisms on two human chromosomal segments. The blocked segment shown here is inverted in a significant fraction of the human population.

Computational Background. This summary is not exhaustive and we focus on the work that is relevant to this paper. Loosely speaking there are two computational approaches to studying the evolutionary relationships of genomes, one of studying the individual gene sequences and the other of studying the arrangement of multiple genes on the genome. A very large amount of literature exists for the first approach (including sequences under the character or feature model), which we will not discuss here to avoid digression. The second approach was initiated by Sankoff [30]: the description of chromosomal inversions in Drosophila had appeared way back in early part of last century [34]. An active interest has been taken in the study of genome rearrangements in the last decade resulting in some very interesting observations and debates in the community.

In the context of genome rearrangements, genomes are viewed as permutations where each integer corresponds to a unique gene or marker. For monochromosomal genomes, the most common rearrangement is *inversion* that is often called *reversal* in the area of bioinformatics. Without loss of generality a permutation of length n with $i \leq j$, can be written as π_1, the inversion on π_1 defined as $r^{ij}(\pi_1)$ and the transposition on π_1 defined as $t^{ijk}(\pi_1)$ where the reversed or transposed segment is underlined.

$$\pi_1 = p_1 p_2 \dots p_{i-1} \underline{p_i p_{i+1} p_{i+2} \dots p_j} p_{j+1} \dots p_k p_{k+1} \dots p_n$$
$$r^{ij}(\pi_1) = p_1 p_2 \dots p_{i-1} \underline{p_j p_{j-1} \dots p_i} p_{j+1} \dots p_k p_{k+1} \dots p_n$$
$$t^{ijk}(\pi_1) = p_1 p_2 \dots p_{i-1} p_{j+1} p_{j+2} \dots p_k \underline{p_i p_{i+1} \dots p_j} p_{k+1} \dots p_n$$

Clearly, $r^{ij}(r^{ji}(\pi)) = \pi$ leading to the idea of a shortest inversion path between two permutations. This shortest inversion path between π_1 and π_2 is the distance between the two given as $D^r(\pi_1, \pi_2)$. However, computing $D^r(\pi_1, \pi_2)$

[3] Transposition is the process in which a *transposable element* is removed from one site and inserted into a second site in the DNA.

for a a given pair of permutations π_1 and π_2 is NP-complete [6]. Hannanhelli and Pevzner showed that by supplementing the genes with signs, this problem could be solved in polynomial time by using graph structures termed *hurdles* and *fortresses* [23]: this is perhaps the most cited work in the area of computational genome rearrangements. The central idea has been subsequently conceptually simplified using common permutation patterns (called *intervals*) and PQ structures [1, 2, 37].

In sequences, the problems of (1) multiple sequence alignment and (2) the construction of the implicit phylogeny tree, have been traditionally separated for simplicity [22, 20, 35]. Such a distinction under the genome rearrangement model is not so obvious. However breakpoint phylogeny was introduced by Sankoff and Blanchette [31] to study this problem under a simplified cost function of minimizing the number of breakpoints. Heuristic approaches also have been applied to this problem in [32, 4]. A rich body of literature on inferring phylogenies under the sequence or character models exists including attempts at using sequence and distance based methods to genome rearrangement problems [15, 27].

Contributions of this paper. In this paper we present a simple computational model of the multiple genome rearrangement problem. Since the motivation is from ordered chromosomal segments, we deal with *unsigned permutations*. Further, since the inversions and transpositions are within the same species, the distance between the members is observed to be very small.

The central idea of the paper is based on the notion of minimal consensus PQ tree T of permutations introduced in [29] and the fact that the number and size of each (excluding leaf nodes) is $\mathcal{O}(1)$ for a small distance between permutations (Theorem 2). We also propose an annotation scheme (called oriented PQ or oPQ tree), that helps uniquely reconstruct the permutations from the tree. Based on this we pose the problem as a permutation tree construction task, show that this is statistically well-defined (Theorem 7) and propose a simple branch-and-bound solution. The scheme produces the phylogeny tree as well as reconstructs all the common ancestors.

Roadmap. In the next section we discuss the PQ data structure and the variants that we propose for the problem. In Section 3, we describe the permutation tree reconstruction problem and an efficient algorithm to compute the phylogeny tree. In Section 5 we discuss the task of including mutations in the problem model and conclude in Section 6.

2 PQ Structures

In this section we discuss PQ structures in forms that will enable us to use them for ancestor reconstruction. We begin with a quick overview of earlier work and then present our modification to the PQ structure.

A PQ structure is a rooted tree with labeled leaves, whose internal nodes are of two types: P and Q. The children of a P-node occur in no particular order

while those of a Q-node appear in a left-to-right or right-to-left order [25]. We designate a P-node by a circle and a Q-node by a rectangle. Two PQ trees T and T' are equivalent, denoted $T \equiv T'$, if one can be obtained from the other by applying a sequence of the following transformation rules: (1) arbitrarily permute the children of a P-node, and (2) reverse the children of a Q-node. A *frontier* of tree T, $F(T)$, is the sequence of leaf nodes in the left to right order. For example, in Figure 2, $F(T_1) = F(T_2) = 0123456789$. $\mathcal{C}(T)$ is defined as $\mathcal{C}(T) = \{F(T')|T' \equiv T\}$.

Next, consider a sequence s of length n defined on a finite alphabet Σ. A permutation pattern p on s is defined as a set of characters $\sigma_i \in \Sigma$, that appear possibly in different orders at different locations in the input [8]. For example, let $s = \sigma_1\sigma_4\sigma_2\sigma_3 \ldots \sigma_1\sigma_2\sigma_3\sigma_4$. Then $\pi = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ is a *permutation pattern* that appears at the beginning and end of s. Π denotes a collection of permutations (patterns) π_i. In [8], a notion of *maximality* of permutation patterns was used (we do not reproduce the formal definition of *occurrence* here and instead appeal to the reader's intuition):

Definition 1. ***(maximal) [8]*** *A permutation pattern π_1 is non-maximal with respect to π_2, if each occurrence of π_1 is covered by an occurrence of π_2 and each occurrence of π_2 covers an occurrence of π_1.*

The notation for maximality was later shown to have the same structure as PQ trees and the idea of a minimal consensus PQ tree of a collection of permutations was introduced in [29]:

Definition 2. ***(minimal consensus PQ tree** $T(\Pi)$**)*** *[29] Given Π, a consensus PQ tree T of Π, written as $T(\Pi)$, is such that $\Pi \subseteq \mathcal{C}(T)$ and the consensus PQ tree is minimal when there exists no $T' \not\equiv T$ such that $\Pi \subseteq \mathcal{C}(T')$ and $|\mathcal{C}(T')| < |\mathcal{C}(T)|$.*

We introduce a modification to the PQ tree that will enable us to reconstruct two permutations π_1 and π_2, from their minimal consensus PQ tree.

Definition 3. ***(oriented PQ tree, oPQ** $\boldsymbol{T}_{\pi_1}(\pi_2)$**)*** *Given two permutations π_1 and π_2, consider $T' \in T(\{\pi_1, \pi_2\})$ (Definition 2) with $\pi_1 = F(T')$). The oriented PQ (oPQ) tree $\boldsymbol{T}_{\pi_1}(\pi_2)$ is defined by annotating T' as follows. Each Q node is annotated with ($\longrightarrow$) or ($\longleftarrow$) labels: the ($\longrightarrow$) label indicates that the two segments are identical in π_1 and π_2 and ($\longleftarrow$) label indicates that the two segments are flipped. The k children of a P node are numbered by integers 1 to k denoting the order in which they appear in π_2 (they appear as 1, 2, ..., k in π_1).*

Figure 2 shows an example of how an oriented PQ (oPQ) tree succinctly describes a pair of permutations. A *frontier*, $F(\boldsymbol{T})$, of an oriented tree $\boldsymbol{T}$ is simply the in-order notation of the PQ tree *excluding* the labeled leafnodes, with the orientation of the Q nodes denoted by a left or right arrow. Further, two oriented trees $\boldsymbol{T}$ and $\boldsymbol{T}'$ are equivalent, denoted as $\boldsymbol{T} \equiv \boldsymbol{T}'$, if and only if $F(\boldsymbol{T}) = F(\boldsymbol{T}')$. Notice that the leaf nodes (which are labeled 0-9 in Figure 2) are ignored while checking the equivalency of oPQ trees. The size of $\boldsymbol{T}$, denoted as, $Size(\boldsymbol{T})$ is the

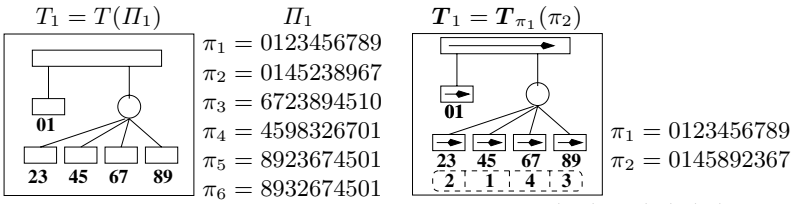


Fig. 2. (a) A minimal consensus PQ tree. (b) The oPQ tree of π_1 and π_2. Note that $Size(T_1) = Size(\boldsymbol{T}_1) = 7$

number of P/Q nodes. The size of T is similarly defined. See Figure 2 for an example of frontier and size of an oPQ tree.

2.1 Algorithmic Implications

Next we explore the time to construct these oPQ trees and the following is a direct consequence of the algorithm described in [29].

Theorem 1 ([29]). *Given two permutations π_1, π_2 of length n each, $\boldsymbol{T}_{\pi_1}(\pi_2)$ is unique. Also, $Size(\boldsymbol{T}_{\pi_1}(\pi_2)) = \mathcal{O}(n)$, and, it can be constructed in $\mathcal{O}(n)$ time.*

Outline of the proof: The algorithm described in [29] constructs $T(\{\pi_1, \pi_2\})$, the minimal consensus PQ tree of π_1 and π_2 in $\mathcal{O}(n)$ time. However, this can be very simply modified to incorporate the orientation in the Q node and the order of the children of the P node to give the oPQ tree $\boldsymbol{T}_{\pi_1}(\pi_2)$. □

Recall that $D^r(\pi_1, \pi_2)$ denotes the inversion distance between π_1 and π_2. Let $D^t(\pi_1, \pi_2)$ denote the shortest transposition path between the two and let $D(\pi_1, \pi_2)$ denote the shortest number of operations, inversion or transposition, that takes π_1 to π_2. The following theorem is central to the proposed algorithm.

Theorem 2. *Given a permutation π of size n and a fixed constant c, let a set of non-equivalent oPQ trees S be defined as follows:*

$$S = \{\boldsymbol{T}_\pi(\pi') \mid D(\pi, \pi') = c \text{ and } \pi' \text{ is a permutation of size } n\}$$

Then $|S| = \mathcal{O}(1)$, and for each $\boldsymbol{T} \in S$, $Size(\boldsymbol{T}) = \mathcal{O}(1)$.

Proof: We first describe the idea of reducing a permutation π (of size n) to a blocked permutation, denoted as $B(\pi)$, which a sequence of b *blocks* where $b \leq n$. Given a pair of permutations, the two can be converted to a sequence of blocks as described below.

Let I_n be the identity permutation of size n and $\pi_1 = I_n$ with π_2 renumbered accordingly. A block is the longest consecutive run of integers, ascending or descending, in π_2. The block is numbered according to its position in $\pi_1 = I_n$. An ascending run block is positively signed and descending run is negatively signed. When the number of blocks is b, then $B(\pi_1{=}I_n) = I_b = (+1)(+2)\ldots(+b)$. The number of blocks is the size denoted by $|B(\pi)|$ and $|B(\pi_1)| = |B(\pi_2)| = b$.

Consider the following example:

$\pi_2 =$ [1 2 3][7 6 5 4][10 11 12][9 8][13 14 15 16] $\Longrightarrow$ +1 -2 +4 -3 +5

$\pi_1 =$ [1 2 3][4 5 6 7][8 9][10 11 12][13 14 15 16] $\Longrightarrow$ +1 +2 +3 +4 +5

Thus $B(\pi_1{=}I_n) = (+1)(+2)(+3)(+4)(+5)$ and $B(\pi_2) = (+1)(-2)(+4)(-3)(+5)$. Further, $|B(\pi_1)| = |B(\pi_2)| = 5$.

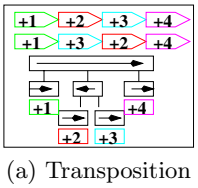
Also, $\boldsymbol{T}_{\pi_1}(\pi_2)$ can be constructed from $\boldsymbol{T}_{B(\pi_1)}(B(\pi_2)))$ where each leaf node block can be replaced by a Q node with the original elements as the ordered children of that node. For instance, in the last example, leaf node corresponding to block 3 has three children 9, 10 , 11 in that order. Thus $Size(\boldsymbol{T}_{\pi_1}(\pi_2)) = \mathcal{O}(b)$.

Lemma 1. *Given two permutations π_1, π_2 of length n each, if $D(\pi_1, \pi_2) = c$, a constant, then $Size(\boldsymbol{T}_{\pi_1}(\pi_2)) = \mathcal{O}(1)$.*

Outline of the proof: Let $\pi_1 = I_n$ and π_2 be renumbered accordingly. Let b be the size of the blocked permutations $B(I_n)$ and $B(\pi_2)$. We need to estimate b for a constant c. Consider the case $c = 1$. Let $S_{t=1} = \{\pi \mid D^t(I_n, \pi) = 1\}$. Then using Figure 3(a), it is easy to verify that $|B(\pi \in S_{t=1})| \leq 4$. Similarly, let $S_{r=1} = \{\pi \mid D^r(I_n, \pi) = 1\}$. Then, using Figure 3(b), it is easy to verify that $|B(\pi \in S_{r=1})| \leq 3$. Consider the case $c = 2$. Let $S_{r=2} = \{\pi \mid D^r(I_n, \pi) = 2\}$. Then, using Figure 4, it is easy to verify that $|B(\pi \in S_{r=2})| \leq 5$. Enumerating the cases in this manner, it can be verified that b is independent of n and only dependent on c, which is a constant. Thus $Size(\boldsymbol{T}_{\pi_1}(\pi_2)) = \mathcal{O}(1)$. □

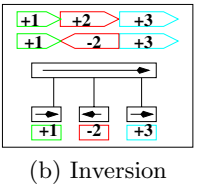
Lemma 2. *Let $B(\pi)$ be the blocked permutation of π of length n. Then $|\{B(\pi) \mid D(I_n, \pi) = c$ and π is a permutation of size $n\}| = \mathcal{O}(1)$.*

Outline of the proof: This follows the lines of the proof of Theorem 2. Consider $c = 1$ with transposition. Figure 3(a) shows one configuration (i.e., $(+1)(+3)(+2)(+4)$) and the other possible (trivial) configurations can be obtained by deleting one or more of blocks (say, $(+3)(+2)(+4){=}(+2)(+1)(+3)$, or, $(+1)(+3)(+2)$). The number of configurations is independent of n and depends only on c. Consider $c = 1$ with inversion. Figure 3(b) shows one configuration (i.e., $(+1)(-2)(+3)$) and the other possible (trivial) configurations can be obtained by deleting one or more of blocks (say, $(-2)(+3){=}(-1)(+2)$, or, $(+1)(-2)$). Thus it can be verified that there are $\mathcal{O}(1)$ distinct blocked permutations for this case as well. Thus using a case-by-case analysis it can be verified that there are only $\mathcal{O}(1)$ distinct configurations when c is a constant. □



(a) Transposition

Treating the blocks as symbols, the left gives $\pi_1 = I_4 = 1234$ and $\pi_2 = 1324$, with $\boldsymbol{T}_{I_4}(\pi_2)$ as shown.
Similarly, in the right, $\pi_1 = I_3 = 123$ and $\pi_2 = 1(-2)3$, with $\boldsymbol{T}_{I_3}(\pi_2)$ as shown.



(b) Inversion

Fig. 3. The only possible configuration for (a) one transposition, (b) one inversion

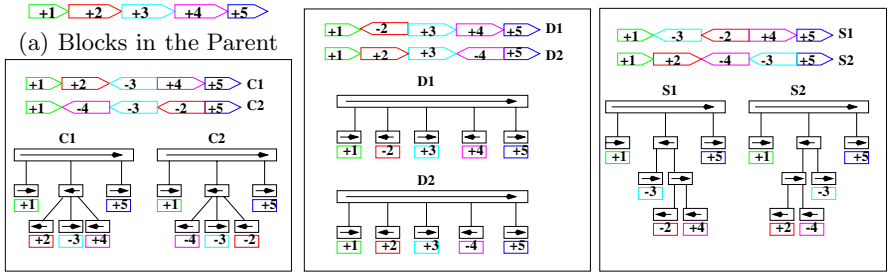


Fig. 4. (a) The parent sequence blocks. The only three possible configurations of two inversion operations are shown in (b)-(d). (b) The two blocks, marked 3 and 2-4 on the parent are *nested*. Labeling the two resulting permutations as $C1$ and $C2$, the first is $\boldsymbol{T}_{C1}(C2)$ (labeled as C1) and the second is $\boldsymbol{T}_{C2}(C1)$ (labeled as C2). (c) The two blocks marked 2 and 4 on the parent are *disjoint*. Labeling the two resulting permutations as $D1$ and $D2$, the first is $\boldsymbol{T}_{D1}(D2)$ (labeled D1) and the second is $\boldsymbol{T}_{D2}(D1)$ (labeled D2). (d) The two blocks, marked 2-3 and 3-4 on the parent, *straddle*. Labeling the two resulting permutations as $S1$ and $S2$, the first is $\boldsymbol{T}_{S1}(S2)$ (labeled S1) and the second is $\boldsymbol{T}_{S2}(S1)$ (labeled S2).

Back to the proof of Theorem 2. For a given permutation π, the number of distinct blocked permutations of π', at distance c from π is only $\mathcal{O}(1)$ by Lemma 2. Each distinct blocked permutation can possibly lead to a distinct oriented PQ tree. Thus the number of such trees is $\mathcal{O}(1)$. And, by Lemma 1, each tree is of size $\mathcal{O}(1)$. □

2.2 Ancestor Reconstruction Through oPQ Templates

Let the *parents* of a set of permutations Π, denoted by $P_c(\Pi)$ be defined as follows: $(\pi' \notin \Pi) \in P_c(\Pi)$, is a permutation such that for each $\pi \in \Pi$, $D(\pi, \pi') \leq c$ for some integer $c \geq 0$. Consider the task of computing $P_c(\Pi)$ where $\Pi = \{\pi_1, \pi_2\}$. If $D(\pi_1, \pi_2) = c$, then for each $\pi \in P_c(\Pi)$, $D(\pi_1, \pi) = c_i$ and $D(\pi_2, \pi) = c - c_i$, for some $0 \leq c_i \leq c$. We first estimate $|P_c(\Pi)|$ in the following theorem.

Theorem 3. *Given $\pi_1 \neq \pi_2$, and a constant c, $|P_c(\pi_1, \pi_2)| = \mathcal{O}(1)$.*

Outline of the proof: The above is a direct consequence of Theorem 2: the parents can be computed from distinct configurations which are $\mathcal{O}(1)$ in number. Further each can give rise to only $\mathcal{O}(1)$ parents, hence the result. □

We next consider the task of reconstructing parent (ancestor) permutations. We do this by computing the oriented minimal consensus PQ trees of a pair of permutations π_1 and π_2 and comparing to oPQ templates. In Figures 3, 4, and 7 treating the blocks as new symbols, we can construct the consensus oPQ trees as illustrated.

Definition 4. ***(oPQ template $\boldsymbol{T}_{\pi_1}(\pi_2)$)*** *An oPQ template is an oPQ tree where each internal node has at most one leaf node. Each leaf node is labeled*

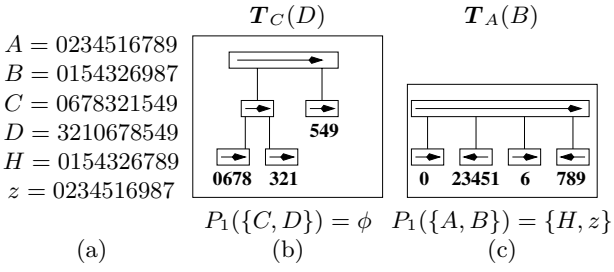


Fig. 5. (a) The permutations and their labels. The oPQ tree $\boldsymbol{T}_C(D)$ in (b) do not match any templates. However the oPQ tree $\boldsymbol{T}_A(B)$ in (c) matches template D1 (or D2) of Figure 4. Consider one of the matches: the first three blocks, marked +1, +2 and +3 (0, 23451, 6) are placed in the same order and the fourth block, marked -4 (789) is reversed giving a parent $z = 0234516987$.

with a signed number i where it represents the ith block in π_1. A plus sign indicates that the block is identical in π_1 and π_2. A minus sign indicates that it is flipped in π_2.

We illustrate oPQ templates in Figure 4: we show the only possible three cases. Since the oPQ's $\boldsymbol{T}$ can be mechanically compared to the consensus oPQ trees of the given permutations, these are called templates. $\boldsymbol{T}_{\pi_2}(\pi_1)$. We next give an overview of the use of an oPQ template through two examples in Figure 5(a-c). Thus the template can be used to reconstruct a common parent.

Space constraints, the algorithmic details of the template matching will be presented in the full version of the paper.

3 The Permutation Tree Construction Problem

Given a collection Π of members where each is defined by a sequence of markers, it is a natural question to reconstruct the phylogeny ("evolutionary") tree.

Definition 5. *(permutation tree $\mathcal{T}$) Given Π a collection of m permutations on integers $1, 2, \ldots, n$, let $\mathcal{T}(V, E)$ be a labeled tree where each node $v \in V$ is labeled by a permutation on integers $1, 2, \ldots, n$ denoted as $\pi(v)$. Let $V' = \{(v \in V) \mid \pi(v) \in \Pi\}$. $\mathcal{T}(V, E)$ is a permutation tree on Π if the following conditions hold: (1) Each $v \in V'$ is labeled bijectively by the elements of Π, (2) for each leaf node $l \in V$, $\pi(l) \in \Pi$, (3) for each $(v_1 v_2) \in E$, $\pi(v_1) \neq \pi(v_2)$ and (4) an internal node v that has degree < 3, must be such that $\pi(v) \in \Pi$ (no linear chains in the tree, except when each v in the chain is given i.e., $\pi(v) \in \Pi$).*

Notice that the extant member can also be at an intermediate node of the tree unlike in most evolutionary tree construction problems [22] where the extant members can only appear as leaf nodes. Also this permutation tree is unrooted. Further, let the length of the permutation tree $\mathcal{T}(V, E)$ be defined as

$$Len(\mathcal{T}) = \sum_{(v_1 v_2) \in E} D(\pi(v_1), \pi(v_2))$$

Problem 4. *(Permutation Tree Construction -PTC- Problem) Given Π, a collection of m permutations of length n each, the PTC problem is to construct the permutation tree $\mathcal{T}(V, E)$ of minimum length.*

It is unclear if this problem has a polynomial time solution. Perhaps a natural restriction is to use signed permutations, instead of unsigned ones, since there exist polynomial time algorithms to compute $D^r(\pi_1, \pi_2)$ [23,26]. However, it is unclear how the common ancestors can be computed on the phylogeny tree and heuristic methods have been proposed in literature for this problem [32,4]. Since our problem is motivated by STRP's on human chromosomes, it is reasonable to assume that for each $(v_1v_2) \in E$ of $\mathcal{T}(V, E)$, $D(\pi(v_1), \pi(v_2)) \leq c$, for some tiny constant c. We call this the cPTC problem and show that finding the exact solution to the problem is computationally tractable.

Problem 5. *(The cPTC Problem) The PTC problem with the added constraint that for each $(v_1v_2) \in E$, $D(\pi(v_1), \pi(v_2)) \leq c$, for some small constant c.*

Notice that the problem definition allows for multifurcating trees. If the permutation tree exists, then Π is said to be *compatible*. If no such tree exists, the harder problem is to modify Π to make it compatible.

Problem 6. *(The near-cPTC Problem) Given $\Pi = \{\pi_1, \pi_2, \ldots, \pi_m\}$, the near-cPTC problem is to minimize $\sum_{i=1}^{m}(D(\pi_i, (\pi_i' \in \Pi'))$ such that Π' is compatible.*

We claim that the PTC is a reasonable definition of the problem by proving that a randomly generated Π is seldom compatible. We first state two lemmas that lead to the theorem.

Lemma 3. *Given a permutation π of length n, $|\{(\pi' \neq \pi) \mid D(\pi', \pi) = 1\}| < n^q$, for some constant q.*

It is easy to verify that $q = 2$ for the inversion operation and $q = 3$ for the transposition operation.

Lemma 4. *Let Π_i, $1 \leq i \leq k$, be k sets of independently chosen random permutations. If $\pi_i \in P_1(\Pi_i)$ then each π_i is independent of π_j, $i \neq j$.*

Theorem 7. *Given a random collection Π of m permutations of size n each, the expected number of permutation trees on Π is $o(1)$.*

Outline of the proof: Using Lemma 3 we compute the following probability where q is some constant:

$$\mathrm{P}(\pi \text{ is the immediate parent of some } \pi_1 \text{ and } \pi_2) = \frac{n^q}{n!^2}$$

Given a tree $\mathcal{T}$ with m leaf nodes, the probability $\mathrm{P}^{\Pi}(\mathcal{T})$ of it being a permutation tree on Π, $\mathrm{P}^{\Pi}(\mathcal{T}) = \left(\frac{n^q}{n!^2}\right)^m$ by Lemma 4. Further, N, the number of possible trees configurations is (this is the number of strictly bifurcating trees on m leaves and is a lower bound on all possible trees):

$$N \leq \frac{(2m-4)!}{2^{m-2}(m-2)!}$$

By linearity of expectation, the expected number of trees on Π is bounded by $N\mathrm{P}^{\Pi}(\mathcal{T})$, which is $o(1)$ since $m \leq n!$. □

Algorithm. We next give a branch and bound algorithm to solve the cPTC problem (see Problem 5) using oPQ templates (trees). Due to space constraints, the details will be presented in the full version of the paper.
Input: Π, a set of m permutations of size n each.
Output: A minimum length tree $\mathcal{T}(V, E)$ and a mapping $P : (v \in V) \rightarrow \Pi^*$, sending $(v \in V) \mapsto (\pi \in \Pi^*)$, where $\Pi \subset \Pi^*$.

4 Experiments

Here we present the results of a set of preliminary experiments. For comparison purposes, the reference rooted phylogeny tree, $\mathcal{T}_r$, is fixed for 50 experiments. We used two reference trees, one a bifurcating and the other a multi-furcating whose exact topology is shown in Figure 6. Each has 16 leaf nodes and the first has 15 and the second has 10 internal nodes. The branch and bound algorithm is used to construct the tree and the oPQ templates used are shown in Figures (4),(3) and (7).

To generate an experiment, we start with a sequence of length 200. This is deemed to be the root node. We follow the topology of $\mathcal{T}_r$ and produce children of the node by randomly selecting a segment of length 5 to 15 for an evolutionary event. This event is either a transposition or an inversion. For the former, yet another location in the sequence is chosen randomly (also no two children of a node have transpositions). To construct a data set Π, we collect all the 16 leaf nodes and randomly pick 4 to 7 of the internal nodes. In 4 experiments we found that Π admitted multiple phylogenies with up to 3 trees, in the bifurcating phylogeny. In the multifurcating tree Π admitted multiple phylogenies in 6 experiments with up to 4 trees.

Hot spot based model: In a variation to the model, we marked multiple (8 to 12) segments in the sequence as hot pot regions. Zero or one evolutionary event was introduced (picked at random with equal probability) in each of the hot spot segments. The data set Π_i is generated for each hot spot region i by yanking subsequences of length 20 around the regions (thus they are no longer strict permutations). Notice that for the same experiment (i.e the same reference $\mathcal{T}_r$ tree) sometimes $|\Pi_{i_1}| \neq |\Pi_{i_2}|$, for $i_1 \neq i_2$, depending on absence/presence of evolutionary events in that region. We compute a phylogeny tree $\mathcal{T}_i$ for each region i and construct a consensus tree to obtain the resulting phylogeny tree. In 44 of the 50 experiments we constructed a correct phylogeny tree in the bifurcating phylogeny and in 41 of the 50 experiments in the multifurcating phylogeny.

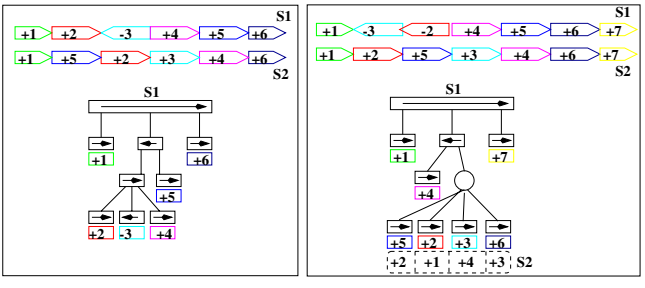
Noisy data: In yet another variation, we introduce random noise to the data. We experiment with four scenarios: (a) inserting and/or deleting random positions, (b) inserting random *segment* of size from 2 to 7, (c) deleting random segments, and, (d) a combination of (b) and (c). The number of trees that are correct out of the 50 are tabulated below.

	(a)	(b)	(c)	(d)
Bifurcating	39	35	31	28
Mulitfurcating	36	28	30	26

Handling missing or spurious data requires extra care and we are looking into probabilistic methods of incorporating this into the model.

(a) bifurcating $\mathcal{T}_r$ (b) multifurcating $\mathcal{T}_r$

Fig. 6. The topology of the reference trees in the simulation experiments



S1: Block 3 is inverted
S2: Block 5 is transposed
(a)

S1: Block 2-3 is inverted
S2: Block 5 is transposed
(b)

S1: Block 2-3 is inverted
S2: Block 5 is transposed
(c)

S1: Block 5 is transposed
S2: Block 6 is transposed
(d)

Fig. 7. In (a) and (c) the parent block is (+1)(+2)(+3)(+4)(+5)(+6) and in (b) and (d) parent block is (+1)(+2)(+3)(+4)(+5)(+6)(+7). Two permutations, S1 and S2, each at exactly one inversion and one transposition distance respectively from the parent. In each, the oPQ template $\boldsymbol{T}_{S1}(S2)$ is displayed with label S1.

5 Mutations

We will assume that the permutation on the markers will also include the specific allelic form it represents, i.e., say the copy number in case of micro satellites and the nucleic acid base in case of SNP's (Single Nucleotide Polymorphism). Let $D^a(\pi_1, \pi_2)$ denote the number of markers that differ in their allelic form. For example, if $\pi_1 = 1^a2^a3^b4^c5^a$, $\pi_2 = 1^a3^a2^a5^c4^c$, where the superscript denotes an encoding of the allelic form, then $D^a(\pi_1, \pi_2) = 2$ since markers 3 and 5 vary in their allelic forms.

The proposed approach can be extended to include mutations and we are currently exploring this direction. In fact, in practice, the problem may be simplified by the use of mutations since, this will help time-order the events. We propose a two-step approach to the problem: first reconstruct the phylogeny tree without using mutations. In the second step the tree can be resolved using the mutation information. The reason for taking this two-step approach is that the assumption that $D^a()$ is small is no longer valid under mutations. We are currently experimenting on synthetic data to validate this approach.

6 Conclusion

Here we present a systematic way of studying large scale genome rearrangements to construct a phylogeny tree. The problem is motivated by the discoveries of large number of inversions and transpositions within the human population. The approach is based on our earlier work on computing minimal consensus PQ trees of permutations along with the observation that when the edit distances are small, only $\mathcal{O}(1)$ number of PQ trees of size $\mathcal{O}(1)$ each need to be considered. This gives an efficient algorithm to compute the underlying phylogeny tree and the reconstruction of all the common ancestors. Our preliminary experiments on simulated data show promising results.

Acknowledgments. My sincere thanks to the anonymous referees who with their careful reading and incisive comments helped improve the paper. I would like to thank Oren and Enam for their efforts with the implementation of the consensus PQ tree construction.

References

1. A. Bergeron. A very elementary presentation of the hannenhalli-pevzner theory. In *Proc. of the Twelfth Symp. on Comp. Pattern Matching*, volume 2089 of *Lecture Notes in Computer Science*, pages 106–117. Springer-Verlag, 2001.
2. A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. In *Proc. of COCOON*, volume 2089 of *Lecture Notes in Computer Science*, pages 68–79. Springer-Verlag, 2003.
3. S. Bloom. Using genetics to unearth our path on earth. *J. of. Clinical Investigation*, 115:1395, 2005.

4. G. Bourque and P. A. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. In *Genome Research*, pages 26–36. Cold Spring Harbor Laboratory Press, 2002.
5. R. L. Cann, M. Stoneking, and A. C. Wilson. Mitochondrial DNA and human evolution. *Nature*, 356:389–390, 1992.
6. A. Caprara. Formulations and complexity of multiple sorting by reversals. In *Proceedings of the Annual Conference on Computational Molecular Biology (RECOMB99)*, pages 84–93. ACM Press, 1999.
7. The Intrnational HapMap Consortium. The International HapMap Project. *Nature*, 426:789–796, 2003.
8. Revital Eres, Gad Landau, and Laxmi Parida. A combinatorial approach to automatic discovery of cluster-patterns. In *Proc. of the Third Wrkshp. on Algorithms in Bioinformatics*, volume 2812 of *Lecture Notes in Bioinformatics*, pages 139–150. Springer-Verlag, 2003.
9. A. Kakizuka et al. Chromosomal translocation t(15;17) in human acute promyelocytic leukemia fuses rar alpha with a novel putative transcription factor, PML. *Cell*, 66(4):663–674, 1991.
10. A. Kakizuka et al. Identification of novel genes, SYT and SSX, involved in the t(X;18)(p11.2;q11.2) translocation found in human synovial sarcoma. *Nature Genetics*, 7:502–508, 1994.
11. C. A. Tilford et. al. A physical map of the human Y chromosome. *Nature*, 409:943–945, 2001.
12. D. Lakich et. al. Inversions disrupting the factor VIII gene are a common casue of severe haemophilia. *Nature Genetics*, 5:236–241, 1993.
13. H. Kehrer-Sawatzki et. al. Molecular characterizations of the pericentric inversion that causes difference between chimpanzee chromosome 19 and human chromosome 17. *Am J. of Hum. Genetics*, 71:375–388, 2002.
14. H. Stefansson et. al. A common inversion under selection in europeans. *Nat. genetics*, 37(2):129–137, 2005.
15. M. E. Cosner et. al. An empirical comparison of phylogenetic methods on chloroplast gene order data in campanulaceae. *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, 2000.
16. M. J. Pettenati et. al. Paracentric inversions in humans: a review of 446 paracentric inversions with presentations of 120 new cases. *Am J. of Med. Genetics*, 55:171–187, 1995.
17. M. L. Bondeson et. al. Inversion of the IDS gene resulting from recombination with IDS-related sequences is a common cause of the Hunter syndrome. *Molecular Genetics*, 4:615–621, 1995.
18. M. T. Pletcher et. al. Use of comparative physical and sequence mapping to annotate mouse chromosome 16 and human chromosome 21. *Genomics*, 74:45–54, 2001.
19. S. Giglio et. al. Olfactory receptor-gene clusters, genomic inversion polymorphisms and common chromosome rearrangements. *Am J. of Hum. Genetics*, 68:874–883, 2001.
20. J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, 2004.
21. S. L. Gersen and M. B. Keagle. *Principles of Clinical Cytogenetics*. 2004. Humana Press.
22. D. Gusfield. *Algorithms on strings, trees and sequencess: computer science and computational biology*. Cambridge University Press, New York, 1997.

23. S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In *J. of ACM*, volume 46, pages 1–27. ACM Press, 1999.
24. M. A. Huynen, B. Snel, and P. Bork. Inversions and the dynamics of eukaryotic gene order. *Trends in Genetics*, 17:304–306, 2001.
25. Booth K. and Leukar G. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.
26. H. Kaplan, R. Shamir, and R. E. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. In *SIAM J. of Computing*, volume 29, pages 880–892, 1999.
27. Md Enamul Karim, Laxmi Parida, and Arun Lakhotia. Using permutation patterns for content-based phylogeny. In *to appear in Lecture Notes in Bioinformatics*, 2006.
28. L.Feuk L, J. R. Macdonald, T. Tang, A. R. Carson AR, M. Li M, G. Rao, R. Khaja, and S. W. Scherer. Discovery of human inversion polymorphisms by comparative analysis of human and chimpanzee dna sequence assemblies. *PLoS Genetics*, 1(4), 2005.
29. Gad Landau, Laxmi Parida, and Oren Weimann. Using pq trees for comparative genomics. In *Proc. of the Symp. on Comp. Pattern Matching*, volume 3537 of *Lecture Notes in Computer Science*, pages 128–143. Springer-Verlag, 2005.
30. D. Sankoff. Edit distance for genome comparison based on non-local operations. In *Proc. of the Third Symp. on Comp. Pattern Matching*, pages 121–135. Springer-Verlag, 1992.
31. D. Sankoff and M. Blanchette. Multiple genome rearrangement and breakpoint phylogeny. *Journal of Computational Biology*, 5(3):555–570, 1998.
32. D. Sankoff, G. Leduc, N. Antoine, B. Paquin, B. Lang, and R. Cedergren. Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome. *Proc. Nat. Acad. Sci.*, 89:6575–6579, 1992.
33. K. Small, J. Iber, and S. T.Warren. Emerin deletion reveals a common X-chromosome inversion mediated by inverted repeats. *Nature Genetics*, 16:96–99, 1997.
34. A. H. Sturtevant. Genetic studies on Drosophila melanogaster. *Genetics*, 5:488–500, 1920.
35. M.S. Waterman. *An Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman Hall, 1995.
36. www.nationalgeographic.com/genographic. 2005.
37. S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

Intron Loss Dynamics in Mammals

Jasmin Coulombe-Huntington and Jacek Majewski

Department of Human Genetics, McGill University, Montreal, Canada
jacek.majewski@mcgill.ca

Abstract. We used whole-genome sequence alignments of human, mouse, rat, and dog to perform a genome-wide analysis of intron loss and gain events in over 17,000 mammalian genes. We found no evidence for intron gain and 122 cases of intron loss, most of which occurred within the rodent lineage. Majority (68%) of the deleted introns were extremely small (<150 bp), significantly smaller than average. The intron losses occurred almost exclusively within highly expressed, housekeeping genes, supporting the hypothesis that intron loss is mediated via germline recombination with spliced mRNA intermediates. This study constitutes the largest scale analysis for intron dynamics in vertebrates to date and allows us to confirm and extend several hypotheses previously based on much smaller samples.

Keywords: comparative genomics, gene structure, introns, mammalian evolution.

1 Introduction

Reconstructing the evolutionary history of spliceosomal introns remains one of the most fervently debated topics in eukaryotic evolution. The long-standing debate over introns early versus introns late [1, 2] contrasts the ideas of introns either originating in the early RNA world, or evolving from an expansion of Type-2-like self-splicing introns in an early eukaryotic ancestor [3]. Understanding the natural history of introns is essential to understanding their function: are introns simply selfish DNA elements that have been maintained in large genomes akin to retrotransposons, or do they serve a function, such a permitting exon shuffling, and alternative splicing, resulting in increased proteome complexity?

Evolutionary investigations of the dynamics of intron gains and losses are generally hampered by the limited availability of high quality data on the sequence and structure of gene orthologues from diverse species. To date, we have been unable to utilize the entire gene complements of most organisms in question, and the datasets commonly used range from hundreds [4] to at most one thousand genes [5].

Here, we make use of the complete, high quality genomic sequences of four mammalian species: human, mouse, rat, and dog, to investigate intron gain and loss dynamics in mammals. We utilize a gene mapping technique to map annotated reference human genes onto the genome-wide, multi-species sequence alignments, allowing us to investigate the predicted intron-exon boundaries of 152,146 introns within 17,242 autosomal genes. A recent study which considered a much smaller number of mammalian genes [5] uncovered 6 differences in intron positions between

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 156 – 170, 2006.

human and rodents, and suggested that there is no evidence for intron gain, and a very slow rate of intron loss in mammals. Here, we detect over 100 cases of intron loss and still no evidence for any intron gain during mammalian evolution. Our large sample size allows us to determine the relative rates of intron losses in mammalian lineages and characterize the types of introns and genes that appear susceptible to loss.

2 Materials and Methods

DNA Sequences and Interspecies Alignments. We used the RefSeq annotation of the human genomic sequence to extract coding sequences of human genes [6]. Only the sequences which could be in silico translated into their predicted protein were retained. This strategy resulted in a high confidence, non-redundant dataset of 17,242 human autosomal genes, containing 152,146 distinct introns within their coding sequence. We based our analysis on the four available highest-quality mammalian genome assemblies: human (hg17), mouse (mm7), rat (rn3), and dog (canFam2). We mapped the well annotated human genes onto the genome-wide alignments present within the 17-way MultiZ [7] alignment tracks in order to determine the intron-exon structures in the target species. We considered only introns that were flanked by coding, or partially coding, exons, since non-coding UTR sequences are poorly conserved (and often not conserved) among species, and provide poor anchors for detecting splice sites within alignments. We also performed the reverse analysis by mapping a set of 16,068 mouse RefSeq genes (129,336 CDS introns) onto the mouse vs. human genomic sequence alignments.

We used the following criteria to detect intron loss events in the target sequence (or gain in the reference sequence): 1) for each reference species intron we identified the positions of both the donor and acceptor splice sites; 2) within the target species, we flagged an intron as potentially lost if the distance between the donor and acceptor sites was lower than a predetermined cutoff, which in the final analysis we set as 25 bp. The latter condition was necessary since alignments are often imperfect at the exon-intron boundary (Figure 1). Especially in the case of intron loss events, the last two base pairs of an exon, which have an AG consensus, tend to align with the downstream intronic acceptor site (also AG), and more serious misalignments are also common. Nevertheless, allowing a margin of 25 bp did not introduce any false positive results (as manually verified in the final curated results), since sequences shorter than 25 bp cannot be efficiently spliced in mammals [8] and correspond to imperfect alignments, rather than actual introns.

Since the genome assemblies and the resulting alignment contain numerous sequencing, assembly, and alignment artifacts, all potential intron loss events were further filtered based on the quality of the underlying alignment. In the process of constructing the BlastZ alignments, gaps in the sequences may be filled in using secondary (non-syntenic) sequences. This significantly increases the proportion of aligned sequences but also results in an increased probability of introducing alignment errors. Thus, only potential intron loss cases which mapped to the highest confidence, top, syntenic, long (encompassing several neighboring genes) nets [9] were retained

for further analysis. Cases occurring in genes which were aligned to multiple or non-syntenic portions of target genomes, which could potentially constitute alignments to duplicate genes or pseudogenes, were rejected.

For all candidate intron losses, we extracted the sequence of 100 bp flanking the intronic site from the genomic sequence assembly and used ClustalW [10] with high gap opening penalty (80) and low gap extension penalty (0) to align it to the human intron-containing sequence, and visualize the detailed evidence for intron loss. After performing some minor supervised adjustments, mainly correcting the misalignment of the terminal AG of an upstream exon with the downstream acceptor site (see above), this allowed us to confirm the deletion events and demonstrate that essentially all of the events are cases of exact deletion, with no alteration to the coding sequence.

Characterization of Genes Involved in Loss Events. In order to functionally classify the genes involved in intron loss events, we used the EASE [11] interface to the Gene Ontology annotation. We identified the GO categories with the highest support – lowest EASE score – for over-representation by the genes within our list, as compared to all known genes.

In order to approximate expression levels and expression breadth of the genes, we used microarray expression data from SymAtlas [12]. Although the relevant variable is the expression level in the germline, this information is currently not available. As a proxy for gene expression levels, we used the mean values of gcRMA summaries across all tissues studied. As an estimate of expression breadth, we used the present/absent calls from the MASS 5.0 summaries and, for each gene, calculated the percentage of tissues where expression was detected.

3 Results

We used the mapping of annotated human exon-intron boundaries onto the mouse, rat and dog genomes to detect changes in gene architecture that occurred during the evolution of the four mammalian species. This approach makes use of the highest quality gene annotation (17242 human genes), but it allows us only to detect either intron loss events that occurred in rodent and dog, or intron gain events that occurred in the human lineage. Thus, we also used the reverse approach: mapping known mouse genes onto mouse/human whole genome alignments. The latter strategy results in a slightly smaller dataset (16068 mouse genes) but allows us to detect intron losses in the human, and intron gains in the mouse genome. This method allowed the mapping of 146964, 141942 and 146727 splice site pairs from human genes to the mouse, rat and dog genomes respectively and 124474 from mouse genes onto the human genome. The results of the combined analyses are listed in Tables 1 and 2. The name and symbol correspond to the human RefSeq gene where the loss/gain event occurred, except for the events in human, where the symbol refers to the mouse gene. The length for dog, mouse and rat events is the length of the corresponding intron in human. For human events, the length is that of the corresponding intron in mouse. We divided the results into isolated events (Table 1), i.e. those where a single intron gain/loss event (or multiple non-consecutive events) occurred in a gene, and concerted events (Table 2) where the change involved multiple successive introns

from the same gene. We propose that the single and multiple events may be mediated by slightly different mechanisms (see Discussion), and the two classes were henceforth analyzed separately.

Table 1. Independent Intron Deletions

RefSeq	Pos	Size	Loss	Symbol
NM_012207	7	275	Dog	HNRPH3
NM_025234	6	238	Dog	WDR61
NM_018445	4	96	Dog	SELS
NM_032259	8	89	Dog	WDR24
NM_004104	42	76	Dog	FASN
NM_025241	8	86	Dog	UBXD1
NM_002096	8	75	Dog	GTF2F1
NM_182752	1	150	Mouse	FAM79A
NM_003132	5	84	Mouse	SRM
NM_006600	5	94	Mouse	NUDC
NM_007122	9	245	Mouse	USF1
NM_004550	6	261	Mouse	NDUFS2
NM_153188	16	130	Mouse	TNPO1
NM_001090	10	96	Mouse	ABCF1
NM_007355	7	204	Mouse	HSP90AB1
NM_138419	6	5966	Mouse	FAM54A
NM_007189	4	82	Mouse	ABCF2
NM_006421	11	112	Mouse	ARFGEF1
NM_001273	39	175	Mouse	CHD4
NM_006191	8	81	Mouse	PA2G4
NM_004184	8	703	Mouse	WARS
NM_001376	67	95	Mouse	DYNC1H1
NM_014030	13	71	Mouse	GIT1
NM_002230	9	196	Mouse	JUP
NM_002805	5	81	Mouse	PSMC5
NM_020695	14	85	Mouse	REXO1
NM_001961	7	359	Mouse	EEF2
NM_020230	10	82	Mouse	PPAN
NM_001379	37	268	Mouse	DNMT1
NM_005498	4	113	Mouse	AP1M2
NM_032377	2	200	Mouse	ELOF1
NM_000516	9	104	Mouse	GNAS

Table 1. *(continued)*

NM_001670	5	968	Mouse	ARVCF
NM_020755	6	133	Mouse	SERINC1
NM_003086	16	179	Mouse	SNAPC4
NM_002046	4	129	Mouse	GAPDH
NM_005216	9	123	Rat	DDOST
NM_014409	4	7100	Rat	TAF5L
NM_003400	14	85	Rat	XPO1
NM_016516	19	114	Rat	VPS54
NM_001747	9	579	Rat	CAPG
NM_005911	8	635	Rat	MAT2A
NM_014670	7	139	Rat	BZW1
NM_004953	18	125	Rat	EIF4G1
NM_006859	5	82	Rat	LIAS
NM_018115	18	108	Rat	SDAD1
NM_017676	5	99	Rat	FLJ20125
NM_002198	4	109	Rat	IRF1
NM_004381	8	813	Rat	CREBL1
NM_007355	5	136	Rat	HSP90AB1
NM_015153	6	81	Rat	PHF3
NM_000971	4	111	Rat	RPL7
NM_018449	20	716	Rat	UBAP2
NM_001001973	5	111	Rat	ATP5C1
NM_003591	13	104	Rat	CUL2
NM_018237	22	95	Rat	CCAR1
NM_003375	5	88	Rat	VDAC2
NM_001011663	2	87	Rat	PCGF6
NM_005146	11	92	Rat	SART1
NM_006842	20	102	Rat	SF3B2
NM_002898	8	179	Rat	RBMS2
NM_013449	26	115	Rat	BAZ2A
NM_007062	4	100	Rat	PWP1
NM_002271	19	89	Rat	RANBP5
NM_002271	23	84	Rat	RANBP5
NM_007111	6	604	Rat	TFDP1
NM_002892	20	217	Rat	ARID4A
NM_207661	13	1030	Rat	FLJ11806
NM_020990	4	129	Rat	CKMT1B

Table 1. *(continued)*

NM_005926	5	101	Rat	MFAP1
NM_005881	9	80	Rat	BCKDK
NM_000546	6	568	Rat	TP53
NM_001961	13	80	Rat	EEF2
NM_001961	11	1156	Rat	EEF2
NM_182513	3	118	Rat	SPBC24
NM_001436	3	86	Rat	FBL
NM_032034	4	80	Rat	SLC4A11
NM_181801	4	108	Rat	UBE2C
NM_007098	26	4829	Rat	CLTCL1
NM_014303	5	91	Rat	PES1
NM_001379	36	797	Rat	DNMT1
NM_001469	4	264	Rat	XRCC6
NM_024319	1	84	Rodent	C1orf35
NM_016252	7	77	Rodent	BIRC6
NM_014763	1	191	Rodent	MRPL19
NM_145212	5	383	Rodent	MRPL30
NM_006773	6	85	Rodent	DDX18
NM_012290	15	82	Rodent	TLK1
NM_001090	16	142	Rodent	ABCF1
NM_022551	3	81	Rodent	RPS18
NM_001634	6	291	Rodent	AMD1
NM_001010	5	105	Rodent	RPS6
NM_004357	8	104	Rodent	CD151
NM_015104	5	129	Rodent	KIAA0404
NM_020680	3	80	Rodent	SCYL1
NM_000920	14	492	Rodent	PC
NM_001166	2	94	Rodent	BIRC2
NM_002046	3	90	Rodent	GAPDH
NM_002046	6	92	Rodent	GAPDH
NM_053275	3	104	Rodent	RPLP0
NM_001312	2	89	Rodent	CRIP2
NM_001003	3	140	Rodent	RPLP1
NM_002952	4	79	Rodent	RPS2
NM_024860	2	72	Rodent	SETD6
NM_024805	2	619	Rodent	C18orf22

Table 1. *(continued)*

NM_002819	4	83	Rodent	PTBP1
NM_002695	3	803	Rodent	POLR2E
NM_003938	13	70	Rodent	AP3D1
NM_020170	8	428	Rodent	NCLN
NM_003685	12	104	Rodent	KHSRP
NM_032285	1	150	Rodent	MGC3207
NM_003333	4	84	Rodent	UBA52
NM_015965	4	236	Rodent	NDUFA13
NM_000979	5	132	Rodent	RPL18
NM_005560	68	83	Rodent	LAMA5
NM_033405	10	97	Rodent	PRIC285
NM_008084	5	85	Human	LOC14433
NM_145370	4	86	Human	Gps1
NM_031170	7	263	Human	Krt2-8
NM_027350	3	112	Human	Nars

Table 2. Multiple Consecutive Intron Deletions

RefSeq	Pos.	Size	Loss	Symbol
NM_012311	11	3747	Rodents	HsKin17
NM_012311	10	2466	Rodents	HsKin17
NM_012311	9	1166	Rodents	HsKin17
NM_012311	8	2261	Rodents	HsKin17
NM_012311	7	3112	Rodents	HsKin17
NM_012311	6	5485	Rodents	HsKin17
NM_012311	5	859	Rodents	HsKin17
NM_012311	4	3038	Rodents	HsKin17
NM_005926	3	256	Mouse	MFAP1
NM_005926	2	2154	Mouse	MFAP1

We were able to uncover a total of 122 changes: 4 occurring in human, 30 in mouse, 46 in rat, 35 in the rodent lineage, prior to the mouse/rat divergence, and 7 in dog. Remarkably, all of the changes were consistent with an intron loss, rather than a gain, i.e. for each case of a deletion of an intron relative to the reference gene structure (either mouse or human), the annotated intron was present in an earlier diverged organism. The loss of each intron was verified by using dog as the outgroup for changes occurring in human, mouse or rat, and using chicken as the outgroup for changes occurring in dog.

Figure 1 shows an example of an intron missing in rodents displayed in the vertebrate MultiZ alignment track of the UCSC genome browser. This case illustrates common misalignments close to the splice sites, which is the reason we had to allow for a 25 bp window when finding the homologous splice sites. To confirm each loss event, we extracted the original genomic sequences from the assemblies and used ClustalW to re-align the reference species intron and 100 bp of flanking upstream and downstream sequences with the homologous target species region. Our analysis shows that at least 117 of the detected intron losses are exact. The remaining 3 cases are also likely to be exact losses but fall into regions of relatively poor quality genomic sequence and require single base insertion/deletion events in the alignments.

Fig. 1. Genome Browser multi-species alignments from the UCSC website for the *DDX18* intron lost in rodents. Uppercase, boxed sequences correspond to exons.

Rates of Intron Loss/Gain. We find a very low rate of intron loss throughout the mammalian evolution and no evidence for intron gain. Based on the total number of donor/acceptor splice site pairs identified in the alignments, we determined the rates of intron loss per million years per intron as: $5.32 \cdot 10^{-6}$ for the mouse-rat common ancestor, $6.58 \cdot 10^{-6}$ for mouse, $1.08 \cdot 10^{-5}$ for rat, $5.30 \cdot 10^{-7}$ for dog and $4.28 \cdot 10^{-7}$ for man. These estimates assume that human and dog lineages diverged 95 MYA, human and rodent 75 MYA [13], and mouse and rat 30 MYA [14, 15]. In order to assess whether the rates are proportional to generation time, we used the age of sexual maturity of each organism – 1/6, 1/3, 3, and 12 years for mouse, rat , dog and man, respectively – to calculate a Pearson correlation coefficient. Although, the relevant

generation time is that of the ancestral species in which the deletions actually occurred, but it is reasonable to assume that the average generation time of a specific lineage should be proportional to that of its descendant species. We obtain a negative correlation of -0.71 (p [one-tailed] = 0.15), as we would expect. This correlation is not significant possibly because there are not enough species in the analysis. It is also possible that generation time is not the only factor affecting the rate of intron loss. Other possible factors may include the effective population size of each lineage, which could lead to differences in the probabilities of fixation of the changes.

Sizes of Deleted Introns. One of the most striking characteristics distinguishing the deleted introns from the norm was their extremely small size. The average size of a human intron is 6259 bp, while the deleted cases were on average 355 bases long in human. Figure 1 illustrates the difference in size distribution of deleted introns and that of all introns. The difference in the distributions is highly statistically significant as measured by a two-sample t-test assuming unequal variance (t = 57.3, df = 208, p [two-tailed] < 10^{-10}). Most of the deleted introns (78 out of 116) are smaller than 150 bases. We further investigated five cases of unusual intron deletions that exceeded 1000 bp in length (5968, 7100, 1030, 1158, and 4380 nucleotides in genes FAM54A, TAF5L, FLJ11806, EEF2, CLTCL1, respectively). Four of those cases occurred in the rat lineage and one in mouse. We identified the corresponding intron in the closest relative (mouse, when the loss occurred in rat, and vice versa) and observed that the introns in the closest relative were actually considerably shorter than in human (984, 4902, 224, 134, and 79 bases respectively) and hence were likely to be short at the time of the deletion. This suggests that the size of the intron must be an important factor affecting the underlying molecular mechanism of deletion.

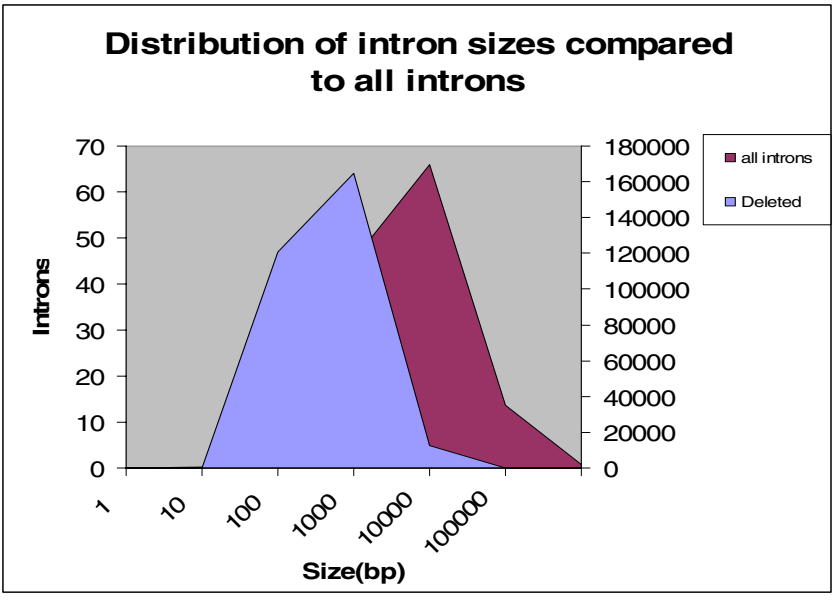


Fig. 2. Log- size distribution of deleted introns versus all introns

Intron Phases. Introns can be classified as phase 0 (inserted between two codons), phase 1 (after the first base of a codon), or phase 2 (after the second base). We examined the phase distribution of the 116 deleted introns from Table 1 and compared it to the phase distribution of all introns from the RefSeq dataset. The proportions for the deleted introns were 0.52, 0.26 and 0.22 for phases 0, 1 and 2 respectively, while the ratios for the control were 0.46, 0.32 and 0.22. The distribution of phases did not differ significantly from the expected ($\chi 2 = 2.24$, df = 2, p = 0.33).

Positions of Deleted Introns within Genes. We wanted to assess whether intron deletions occurred preferentially to one half of the gene. To organize each event as belonging to the 5' or 3' half of the gene, we divided the number of the intron (see table 1) – excluding introns in UTR regions since these were not included in our analysis – by the total number of introns in the gene in order to obtain a fraction. Assuming a null uniform distribution for the positions of deleted introns, we used a $\chi 2$ analysis to compare the proportion of deleted introns that fell in each half of the gene. We found that the distribution of deleted introns was significantly skewed toward the 3' half of the gene ($\chi 2 = 7.76$, df = 1, p = 0.0053).

Splice Site Characteristics. We examined the distributions of bases around both splice sites and compared to the distributions for all introns. We found that the consensus at the 5' splice site was not significantly different from the control. However, at the 3' splice site, the two positions after the AG splice site had a significantly greater frequency of the bases G ($\chi 2 = 3.82$, df = 1, p = 0.05) and T ($\chi 2 = 4.93$, df = 1, p = 0.03), respectively.

Expression and Ontology. We used the EASE [11] interface to classify our genes into GO categories (biological process and by cellular component) and characterize the types of genes that undergo intron deletion events. EASE can be used to calculate over-representation statistics for each GO category (using an EASE score, defined as the upper bound of the Jackknife Fisher exact probability distribution). In Table 3, we list the most over-represented biological processes (EASE score < 0.05). We note that most of the genes with intron deletions are involved in biosynthesis, metabolism, translation, transcription, and RNA processing. Almost all of the overrepresented categories correspond to ubiquitous, housekeeping functions, suggesting that intron deletion events occur predominantly in genes that are both highly expressed AND expressed in the germline. In order to further confirm this hypothesis, we utilized microarray expression data available from SymAtlas [12] to determine the expression intensities and breadths of the candidate genes. Since germline gene expression levels are not known, we used averaged gcRMA (robust multi-array analysis, normalized for GC content) expression over all tissues as a proxy for germline expression, and compared the averages of the intron-deleted sample to all genes. The mean gcRMA expression level was of 952 overall, and significantly higher, 9560, for the genes with an intron deletion, as confirmed by a two-sample t-test assuming unequal variance (t = -4.3, df = 108, p [two-tailed] = 3.58•10-5). In order to study the breadth of expression, we used MASS 5.0 present/absent calls from more than 300 tissues and cell lines and determined the fraction of tissues where expression was positively detected (present). Again, we compared the expression breadth for genes with intron deletions (mean = 0.54) to that of all genes (mean = 0.26) and found a highly significant

difference using the same t-test (t = -6.9, df = 92, p [two-tailed] = 7.15•10-10). Thus intron deletions occur preferentially in genes with housekeeping functions, which have experimentally been determined to be both highly and broadly expressed.

Table 3. Over-represented GO Biological Processes (99 genes categorized in our list as compared to 13802 genes for the control)

GO Biological Process	List Hits	Population Hits	EASE Score
protein biosynthesis	19	650	6.17E-07
biosynthesis	26	1199	6.37E-07
macromolecule biosynthesis	22	1002	5.69E-06
metabolism	75	7637	2.68E-05
translation	9	236	0.000272
Pol II promoter transcription	12	477	0.000557
nucleic acid metabolism	38	3429	0.003043
RNA processing	10	430	0.00346
RNA metabolism	10	460	0.005365
protein metabolism	31	2696	0.005747
nucleocytoplasmic transport	5	108	0.007294
spermine biosynthesis	2	2	0.014151
translational elongation	3	27	0.015607
spermine metabolism	2	3	0.021152
spermidine metabolism	2	3	0.021152
spermidine biosynthesis	2	3	0.021152
intracellular transport	10	613	0.02996
transcription	26	2426	0.030485
polyamine biosynthesis	2	7	0.048667

Case Study: *GAPDH*. We performed a detailed analysis of the *GAPDH* gene, where we found evidence of multiple, independent intron losses occurring in mouse, human, and rat. *GAPDH* is known to be very highly expressed, which follows the premise

that expression in the germline is essential for intron loss. We extracted genomic and mRNA *GAPDH* sequences for 15 vertebrate species, and used multiple alignments to reconstruct the intron/exon structure of the gene in each species (Figure 2). A Dollo parsimony approach (assuming a single appearance of the derived character – intron) suggests that there were no gain events throughout vertebrates, but numerous losses, including several independent losses of the same intron (intron 9 of the ancestral gene). The result also suggests that the phenomenon of intron loss in vertebrates (at least within this gene) may be accelerated in the mammalian branch.

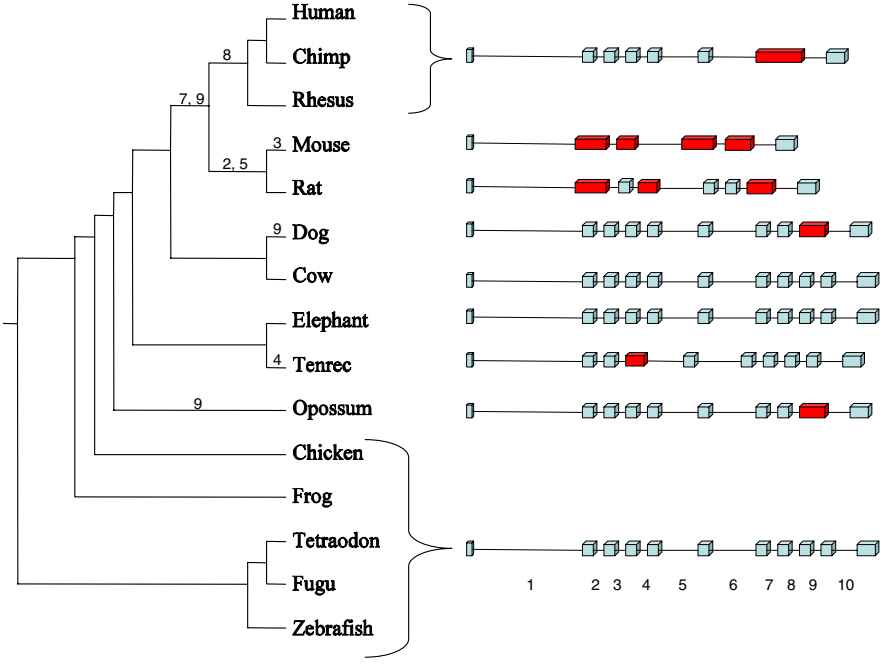


Fig. 3. The evolution of the intron-exon structure of the GAPDH gene throughout the vertebrate phylogeny. The numbers on the branches indicate the inferred deletion events. The introns are numbered according to their position within the ancestral gene.

4 Discussion

We identify over 100 cases of intron loss in the four examined mammalian species. Our approach, based on the mapping of known human genes to whole-genome sequence alignments of multiple species, allows us to utilize the annotation information from well studied model species, such as human and mouse, and predict gene structure in other, relatively poorly annotated species. Using our method, we recover all 6 intron deletion events detected in the smaller scale study of Roy and Gilbert [5], and extend their conclusions regarding the patterns of intron loss in mammals. There are several remarkable characteristics of our dataset: 1) losses appear to occur almost exclusively for small introns; 2) essentially all of our examples of loss

are consistent with an exact deletion event; 3) the loss events are biased towards the 3' half of genes, but can be found at all positions; 4) genes that are associated with intron loss events are generally highly expressed and have housekeeping functions; 5) all of the differences in gene structure are consistent with intron loss events – no detected intron insertions have occurred in human or mouse since their divergence. Below, we discuss some implications of these findings.

Mechanism of Intron Loss. It has been suggested that intron loss may be mediated either by genomic deletion events or recombination of the genomic locus with a reverse-transcribed, processed mRNA molecule of the gene [16]. Our analysis suggests that at least 98% (and possibly all) of the observed deletions are exact. In addition, we do not detect any evidence for inexact deletions, which would retain a small part of the intron or remove parts of neighboring exons. It has been argued [5] that random genomic deletion events would be unlikely to always result in exact intron losses. This is even more evident in our large dataset. It would be extremely unlikely that, if intron loss were generally mediated by random deletions, we would not recover any cases of inexact losses. Even in the presence of purifying selection against such potentially deleterious events, it seems plausible that some minor insertion/deletions of the boundary sequence, particularly ones that do not alter the reading frame, would be evolutionarily neutral. Thus the exact character of the detected intron loss events supports the latter model, namely recombination with an intronless cDNA of the gene.

The small size of the introns provides another insight into the mechanism of loss. It is well documented that genetic recombination events occur less frequently in the presence of mismatches, insertions, or deletions within the recombining substrates [17]. We propose that in the cases of intron loss, recombination with cDNA is much more likely if the introns are small, resulting in a high relative effective proportion of sequence identity.

We also find that genes susceptible to intron loss tend to be involved in housekeeping functions and expressed at relatively high levels. Again, high level of expression most likely results in relatively high levels of reverse-transcribed copies of the gene, leading to an increased probability of recombination. A similar effect has been demonstrated for the frequency of processed pseudogenes [18]. Furthermore, in order for the recombination events to result in intron losses that are transmitted to the next generation and have a chance to increase in frequency in the population, the loss events must occur in the germline, as opposed to somatic cells. Thus, germline expression of the gene would be an essential condition for intron loss. In accordance with this prediction, we find that our intron-deleted dataset is highly enriched in housekeeping (ubiquitously expressed) genes. Thus both the expression levels and the expression patterns of the genes support the recombination-mediated model of intron loss.

Finally, we find that the position of the lost introns is significantly biased towards the 3' ends of the genes. This is in accordance with recent studies of lower eukaryotes [19, 20] and again supports intron loss being mediated by recombination, since reverse transcription of the mRNA is believed to occur preferentially from the 3' end. This may reflect the bias in distribution of intron sizes (first introns are generally longer) and selective pressures against deleting regulatory regions (first introns have a greater conservation across species) [21].

Multiple Successive Intron Losses. We identify two cases of multiple consecutive intron losses (see Table 2), which we classify as a distinct type of event. The introns lost in these events are generally considerably longer than those involved in the usual single-loss cases. We propose that this type of rearrangement event occurs as a result of recombination with nearly complete cDNA, whereas in the more typical cases the recombining cDNAs may be incomplete or fragmented. The overall greater available length of substrate involved in the concerted losses may promote recombination despite the presence of long unmatched intronic regions.

Selection Favoring Intron Loss? The preferential intron loss in highly expressed housekeeping genes is also consistent with selection for transcription efficiency favoring the resulting short transcript. However, most of the deleted introns are extremely short while, selection alone would favor the loss of longer introns. In the example of the GAPDH gene, a loss of an 82 bp intron from a 3783 bp transcript would result in only a very modest 2% decrease in the time of transcription. In comparison, loss of the first intron fully contained within the CDS (~1700 bp) could result in a 45 % reduction. We believe that the availability of cDNA and the length of unmatched intronic sequences in the recombining strands are the primary limiting factors in the process of intron loss. Once the gene conversion event occurs, selection may be an additional force increasing the probability of fixation of such events.

Rates of Intron Loss and Gain. The rate of intron loss in mammals appears extremely slow. The fastest genome-wide rate, in the rat lineage is approximately 1 intron loss per 1.53 million years. We note that the rates are not clocklike and seem correlated with the generation time of each lineage: rodent > dog > human. At the current rate, it would take more than 10^{12} years for the human genome to shed half of its introns. Hence, intron loss/gain does not appear to be a major factor in mammalian evolution.

Since no cases of intron gain were detected, we estimate the process to proceed considerably slower than intron loss. Our approach would allow us to detect intron gain events occurring in the mouse/rodent lineage, after its divergence from the human lineage. Since no cases of intron gain (and 63 losses) were detected, we estimate the process to proceed considerably slower than intron gain. Our observations further support the theory of introns being evolutionarily inert and, having expanded through early eukaryotic genomes (or being inherited through earlier yet ancestors), have been gradually, albeit very slowly, disappearing within mammalian lineages over at least the past 100 million years.

Acknowledgments. This research was supported by funds from the Canadian Institute of Health Research and the Canada Research Chairs program.

References

1. de Souza, S.J., et al., *Toward a resolution of the introns early/late debate: only phase zero introns are correlated with the structure of ancient proteins.* Proc Natl Acad Sci U S A, 1998. **95**(9): p. 5094-9.

2. Stoltzfus, A., *Origin of introns--early or late.* Nature, 1994. **369**(6481): p. 526-7; author reply 527-8.
3. Cavalier-Smith, T., *Intron phylogeny: a new hypothesis.* Trends Genet, 1991. **7**(5): p. 145-8.
4. Rogozin, I.B., et al., *Remarkable interkingdom conservation of intron positions and massive, lineage-specific intron loss and gain in eukaryotic evolution.* Curr Biol, 2003. **13**(17): p. 1512-7.
5. Roy, S.W., A. Fedorov, and W. Gilbert, *Large-scale comparison of intron positions in mammalian genes shows intron loss but no gain.* Proc Natl Acad Sci USA, 2003. **100**(12): p. 7158-62.
6. Hinrichs, A.S., et al., *The UCSC Genome Browser Database: update 2006.* Nucleic Acids Res, 2006. **34**(Database issue): p. D590-8.
7. Blanchette, M., et al., *Aligning multiple genomic sequences with the threaded blockset aligner.* Genome Res, 2004. **14**(4): p. 708-15.
8. Lim, L.P. and C.B. Burge, *A computational analysis of sequence features involved in recognition of short introns.* Proc Natl Acad Sci U S A, 2001. **98**(20): p. 11193-8.
9. Kent, W.J., et al., *Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes.* Proc Natl Acad Sci USA, 2003. **100**(20): p. 11484-9.
10. Thompson, J.D., D.G. Higgins, and T.J. Gibson, *CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.* Nucleic Acids Res, 1994. **22**(22): p. 4673-80.
11. Hosack, D.A., et al., *Identifying biological themes within lists of genes with EASE.* Genome Biol, 2003. **4**(10): p. R70.
12. Su, A.I., et al., *Large-scale analysis of the human and mouse transcriptomes.* Proc Natl Acad Sci U S A, 2002. **99**(7): p. 4465-70.
13. Waterston, R.H., et al., *Initial sequencing and comparative analysis of the mouse genome.* Nature, 2002. **420**(6915): p. 520-62.
14. Nei, M., P. Xu, and G. Glazko, *Estimation of divergence times from multiprotein sequences for a few mammalian species and several distantly related organisms.* Proc Natl Acad Sci U S A, 2001. **98**(5): p. 2497-502.
15. Springer, M.S., et al., *Placental mammal diversification and the Cretaceous-Tertiary boundary.* Proc Natl Acad Sci U S A, 2003. **100**(3): p. 1056-61.
16. Logsdon, J.M., Jr., A. Stoltzfus, and W.F. Doolittle, *Molecular evolution: recent cases of spliceosomal intron gain?* Curr Biol, 1998. **8**(16): p. R560-3.
17. Majewski, J. and F.M. Cohan, *DNA sequence similarity requirements for interspecific recombination in Bacillus.* Genetics, 1999. **153**(4): p. 1525-33.
18. Zhang, Z., et al., *Millions of years of evolution preserved: a comprehensive catalog of the processed pseudogenes in the human genome.* Genome Res, 2003. **13**(12): p. 2541-58.
19. Roy, S.W. and W. Gilbert, *The pattern of intron loss.* Proc Natl Acad Sci U S A, 2005. **102**(3): p. 713-8.
20. Sverdlov, A.V., et al., *Preferential loss and gain of introns in 3' portions of genes suggests a reverse-transcription mechanism of intron insertion.* Gene, 2004. **338**(1): p. 85-91.
21. Majewski, J. and J. Ott, Distribution and characterization of regulatory elements in the human genome. Genome Res, 2002. **12**(12): p. 1827-36.

Finding Maximum Likelihood Indel Scenarios

Abdoulaye Baniré Diallo[1,2], Vladimir Makarenkov[2], and Mathieu Blanchette[1]

[1] McGill Centre for Bioinformatics and School of Computer Science, McGill University, 3775 University Street, Montréal, Québec, H3A 2B4, Canada
[2] Département d'informatique, Université du Québec à Montréal, C.P. 8888, Succ. Centre-Ville, Montréal (Québec), H3C 3P8, Canada

Abstract. Given a multiple alignment of orthologous DNA sequences and a phylogenetic tree for these sequences, we investigate the problem of reconstructing the most likely scenario of insertions and deletions capable of explaining the gaps observed in the alignment. This problem, that we called the Indel Maximum Likelihood Problem (IMLP), is an important step toward the reconstruction of ancestral genomics sequences, and is important for studying evolutionary processes and genome function. We solve the IMLP using a new type of tree hidden Markov model whose states correspond to single-based evolutionary scenarios and transitions model dependencies between neighboring columns. The standard Viterbi and Forward-backward algorithms are optimized to produce the most likely ancestral reconstruction and to compute the level of confidence associated to specific regions of the reconstruction. The method is illustrated on a set of 85kb sequences from eight mammals.

Keywords: Ancestral genome reconstruction; Insertions and deletions; Tree-HMM; Ancestral mammalian genomes.

1 Introduction

It has recently been shown that the phylogeny of eutherian mammals is such that an accurate reconstruction of the genome of an early ancestral mammal is possible [1]. The ancestral genome reconstruction procedure involves several difficult steps, including the identification of orthologous regions in different extant species, ordering of syntenic blocks, multiple alignment of orthologous sequences within each syntenic block, and reconstruction of ancestral sequences for each aligned block. This last step involves the inference of the set of substitutions, insertions, and deletions that have may have produced a given set of multiply-aligned extant sequences. While the problem of reconstructing substitutions scenarios has been well studied (starting with Felsenstein (1981) [7]), the inference of insertions and deletions scenarios has received less attention (but see the seminal contribution of Thorne, Kishino and Felsenstein [19]). The difficulty of the problem is due in large part to the fact that insertions and deletions (indels) often affect several consecutive nucleotides, so the columns of the alignment cannot be treated independently, as opposed to the maximum likelihood problem for substitutions [7]. The reconstruction of the most parsimonious scenario of indels required to explain a given multiple sequence alignment has been

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 171–185, 2006.

shown to be NP-Complete by Chindelevitch *et al.*[5], but good heuristics have been developed by Blanchette *et al.* [1], Chindelevitch *et al.* [5], and Fredslund *et al.* [9].

A maximum likelihood reconstruction would be preferable to a most parsimonious reconstruction because it would be more accurate and would allow to estimate the uncertainty related to certain aspects of the reconstruction. Similarly to statistical alignment approaches [13] (which unfortunately remain too slow for genome-wide reconstructions), we seek to gain a richer insight into ancestral sequences and evolutionary processes. In this paper, we thus focus on the problem we call the *Indels Maximum Likelihood Problem (IMLP)*. It consists of inferring the set of insertions and deletions that has the maximal likelihood, according to some fixed evolutionary parameters, and that could explain the gaps observed in a given alignment. An example of the input and output of this problem are shown in Figure 1. Indel evolutionary scenarios are useful for several other problems such as annotating functional regions of extant genomes, including protein-coding regions [17], RNA genes [15], and other types of functional regions [16].

Here, we start by giving a formal definition of the Indel Maximum Likelihood Problem. To solve the problem, we use a special type of tree Hidden Markov Model, which is a combination of a standard Hidden Markov Model and a phylogenetic tree. We show how the most likely path through the tree-HMM leads to the most likely indel scenario and how a variant of the standard Viterbi algorithm can solve the problem. Although the size of the HMM is exponential in the number of extant species considered, we show how the knowledge given by the phylogenetic tree and the aligned sequences allows the state space of the HMM to be considerably reduced, resulting in a practical, yet exact, algorithm. Our implementation is able to solve large problems on a simple desktop computer and allows for an easy parallelization. Finally, we assess the complexity and accuracy of our algorithm on a multiple alignment of eight orthologous mammalian genomic sequences of $\sim$50kb each.

2 The Indel Maximum Likelihood Problem

In this section we will give a precise definition for the Indel Maximum Likelihood Problem (IMLP). Consider a rooted binary phylogenetic tree $T = (V_T, E_T)$ with branch lengths $\lambda : V_T \rightarrow \mathbb{R}^+$. If n is the number of leaves of T, there are $n-1$ internal nodes and $2n-2$ edges.

Consider a multiple alignment A of n orthologous sequences corresponding to the leaves of the tree T. Since the only evolutionary events of interest here are insertions and deletions, A can be transformed into a binary matrix, where gaps are replaced by 0's and nucleotides by 1's. Let A_x be the row of the binarized alignment corresponding to the sequence at leaf x of T, and let $A_x[i]$ be the binary character at the i-th position of A_x. Assume that the alignment A contains L columns, we add for convenience two extra columns, $A[0]$ and $A[L+1]$, consisting exclusively of 1's.

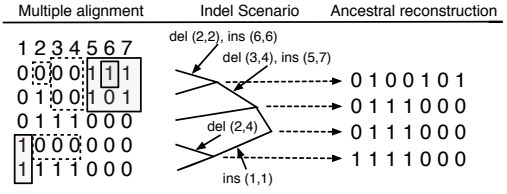


Fig. 1. Example of an input and output to the Indel Maximum Likelihood Problem. The input consists of a multiple alignment (shown on the left in binary format) and the topology and branch lengths of the phylogenetic tree. The output consists of a set of insertions and deletions, placed along the edges of the tree, explaining the gaps (zeros) in the alignment. The dashed boxes in the alignment indicate the deletions and the shaded boxes indicate the insertions of the scenario shown on the right. This set of operations yields the ancestral reconstruction shown on the right.

Definition 1 (Ancestral reconstruction). *Given a multiple alignment A of n extant sequences assigned to the leaves of a tree T, an* ancestral reconstruction *A^* is an extension of A that assigns a sequence $A^*_u \in \{0,1\}^{L+2}$ to each node u of T, and where $A^*_u = A_u$ whenever u is a leaf.*

An ancestral reconstruction thus specifies, for each ancestral node of T, what positions were occupied by a nucleotide and what positions had a gap (see Figure 1 for an example). The following restriction on the set of possible ancestral reconstructions is necessary in some contexts.

Definition 2 (Phylogenetically correct ancestral reconstruction). *An ancestral reconstruction A^* is* phylogenetically correct *if, for any $u, v, w \in V_T$ such that w is located on the path between u and v in T, we have $A^*_u[i] = A^*_v[i] = 1 \implies A^*_w[i] = 1$.*

Requiring an ancestral reconstruction to be phylogenetically correct corresponds to assuming that any two nucleotides that are aligned in A have to be derived from a common ancestor, and thus that all the ancestral nodes between them have to have been a nucleotide. This prohibits aligned nucleotides to be the result of two independent insertions. Assuming that this property holds perfectly for a given alignment A is somewhat unrealistic, but, for mammalian sequences, good alignment heuristics have been developed (e.g. TBA [2], MAVID [3], MLAGAN [4]) and have been shown to be very accurate [2]. In the future, we plan to relax this assumption, but, for now, we will concentrate only on finding phylogenetically correct ancestral reconstructions.

Since we are considering insertions and deletions affecting several consecutive characters, we delimit each operation by the positions s and e in the aligned sequences where it starts and ends. Let x and y be two nodes of the tree, where x is the parent of y. The alignment consisting of rows A^*_x and A^*_y is divided into a set of regions defined as follows (see Figure 2).

Definition 3 (Deletions, Insertions, Conservations, and Length).

- *The region* (s, e) *is a* deletion *if (a) for all* $i \in \{s, \ldots, e\}, A_y^*[i] = 0$*, (b)* $A_x^*[s] = A_x^*[e] = 1$*, and (c) no region* $(s', e') \supset (s, e)$ *is a deletion (i.e. we only consider regions that are maximal).*
- *The region* (s, e) *is an* insertion *if (a) for all* $i \in \{s, \ldots, e\}, A_x^*[i] = 0$*, (b)* $A_y^*[s] = A_y^*[e] = 1$*, and (c) no region* $(s', e') \supset (s, e)$ *is an insertion.*
- *The region* (s, e) *is a conservation if (a) for all* $i \in \{s, \ldots, e\}, A_x^*[i] = A_y^*[i]$ *and (b) no region* $(s', e') \supset (s, e)$ *is a conservation.*
- *The* length *of region* (s, e) *is the number of non-trivial positions it contains:* $l(s, e) = |\{s \leq i \leq e | A_x^*[i] \neq 0 \text{ or } A_y^*[i] \neq 0\}|$.

A pair of binary alignment rows A_x^* and A_y^* can thus be partitioned into a set of non-overlapping insertions, deletions, and conservations.

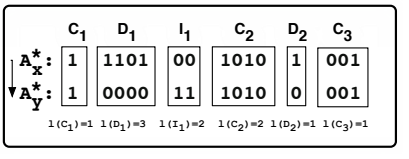


Fig. 2. Example of the partition of a paiwise alignment of A_x^* and A_y^*, where x is the parent of y. The length of each region is given below the region.

Definition 4 (Indel scenario). *The indel scenario defined by an ancestral reconstruction* A^* *is the set of insertions and deletions that occurred between the ancestral reconstructions at adjacent nodes in* T*.*

All that remains is to define an optimization criterion on A^*. Two main choices are possible: a parsimony criterion or a likelihood criterion.

2.1 The Indel Parsimony Problem

The parsimony approach for the indel reconstruction problem has been introduced by Fredslund *et al.* [9] and Blanchette *et al.* [1]. In its simplest version, it attempts to find the phylogenetically correct ancestral reconstruction A^* that minimizes the total number of insertions and deletions defined by A^*:

$$indelParsimony(A^*) = \sum_{u,v:(u,v)\in E} |\{(s, e) : (s, e) \text{ is a deletion or an insertion from } A_u^* \text{ to } A_v^*\}|$$

The Indel Parsimony Problem is NP-Hard [5]. Most authors have studied a weighted version of the IPP where the cost of indels depends linearly on their length (affine gap penalty). Blanchette *et al.* [1] proposed a greedy algorithm, and good exact heuristics have been developed [5,9]. The limitation of these approaches is that they only give a single solution as output, and provide no measure of uncertainty of the various parts of the reconstruction. In contrast, a

likelihood-based approach has the potential of providing a more accurate solution and a richer description of the set of possible solutions.

2.2 Indel Maximum Likelihood Problem

In this section, we define the indel reconstruction problem in a probabilistic framework and similar to the Thorne-Kishino-Felsenstein model [18]. To this end, we need to define the probability of transition between an alignment row A_x^* and its descendant row A_y^*. This probability will be defined as a function of the probability of the insertions, deletions, and conservations that happened from A_x^* to A_y^*.

Let $P_{DelStart}(\lambda(b))$ be the probability that a deletion starts at a given position in the sequence, along a branch b of length $\lambda(b)$, and $P_{InsStart}(\lambda(b))$ is defined similarly. We assume that these probabilities only depend on the length $\lambda(b)$ of the branch b along which they occur, but not on the position where the indel occurs. A reasonable choice is $P_{DelStart}(\lambda(b)) = 1 - e^{-\psi_D \lambda(b)}$ and $P_{InsStart}(\lambda(b)) = 1 - e^{-\psi_I \lambda(b)}$, for some deletion and insertion rate parameters ψ_D and ψ_I, but our algorithm allows for any other choice of these probabilities. Thus, the probability that none of the two events happens at a given position, which we call the probability of a conservation, is given by $P_{Cons}(\lambda(b)) = e^{-(\psi_D+\psi_I)\lambda(b)}$. We assume that the length of a deletion follows a geometric distribution, where the probability of a deletion of length k is $\alpha_D^{k-1}(1-\alpha_D)$ and the probability of an insertion of length k is $\alpha_I^{k-1}(1-\alpha_I)$. One can thus see α_D (resp. α_I) as the probability of extending a deletion (resp. insertion). This assumption, necessary to design a fast algorithm, holds relatively well for short indels, but fails for longer ones [12]. Our algorithm allows the parameters α_D and α_I to depend on the branch b, but the results reported in Section 5 correspond to the case where α_D and α_I were held constant across the tree. The probability that alignment row A_x^* was transformed into alignment row A_y^* along branch b can be defined as follows:

$$\begin{aligned}\Pr(A_y^*|A_x^*, b) = &\prod_{(s,e)\text{: deletion from } A_x^* \text{ to } A_y^*} P_{DelStart}(\lambda(b)) \cdot (\alpha_D^{l(s,e)-1}(1-\alpha_D)) \cdot \\ &\prod_{(s,e)\text{: insertion from } A_x^* \text{ to } A_y^*} P_{InsStart}(\lambda(b)) \cdot (\alpha_I^{l(s,e)-1}(1-\alpha_I)) \cdot \\ &\prod_{(s,e)\text{:conservation from } A_x^* \text{ to } A_y^*} (P_{Cons}(\lambda(b))^{l(s,e)}\end{aligned}$$

This allows us to formulate precisely the problem addressed in this paper:

Indel Maximum Likelihood Problem (IMLP)
Given: A multiple sequence alignment A of n orthologous sequences related by a phylogenetic tree T with branch lengths λ, a probability model for insertions and deletions specifying the values of ψ_D, ψ_I, α_D, and α_I.

Find: A maximum likelihood phylogenetically correct ancestral reconstruction A^* for A, where the likelihood of A^* is:

$$L(A^*) = \prod_{b=(x,y)\in E_T} \Pr[A^*_y | A^*_x, b]$$

3 A Tree Hidden Markov Model

In this section, we describe the tree-based hidden Markov model that is used to solve the IMLP. A tree-Hidden Markov Model (tree-HMM) is a probabilistic model that allows two processes to occur, one in time (related to the sequence history in a given column of A), and one in space (related to the changes toward the neighboring columns). Tree HMMs were introduced by Felsenstein and Churchill [8] and Yang [20] to improve the phylogenetic models that allows for variation among sites in the rate of substitution, and have since then been used for several other purposes (detecting conserved regions [16] and predicting genes [17]). Just as any standard HMM [6], a tree-HMM is defined by three components: the set of states, the set of emission probabilities, and the set of transition probabilities.

3.1 States

Intuitively, each state corresponds to a different single-column indel scenario (although additional complication are described below). Given a rooted binary tree $T = (V_T, E_T)$ with n leaves, each state corresponds to a different labeling of the edges E_T with one of three possible events: I (for insertion), D (for deletion), or C (for conservation). The set $\mathcal{S}$ of possible states of the HMM would then be $\mathcal{S} = \{I, D, C\}^{2n-2}$. However, this definition is not sufficient to model certain biological situations (see Figure 3). We will use the '*' symbol to indicate that, along a certain branch $b = (x, y)$, no event happened because there was a base neither at node x nor at node y. This will happen in two situations: when edge b is a descendant of edge b' that was labeled with D (i.e. the base was deleted higher up the tree), and when there exists an edge b' that is not between b and the root and that is labeled with I (i.e. an insertion happened elsewhere in the tree). The fact that these extraneous events can potentially interrupt ongoing events along branch b means that the HMM needs to have a way to remember what event was actually going on along that branch. This transmission of memory from column to column is achieved by three special labels: I^*, D^*, and C^*, depending on whether the * regions is interrupting an insertion, deletion, or conservation. Thus, we have $\mathcal{S} \subseteq \{I, D, C, I^*, D^*, C^*\}^{2n-2}$. Although this state space appears prohibitively large (6^{2n-2}), the reality is that a number of these states cannot represent actual indel scenarios, and can thus be ignored. The following set of rules specify what states are valid.

Definition 5 (Valid states). *Given a tree $T = (V_T, E_T)$, a state s assigning a label $s(b) \in \{I, D, C, I^*, D^*, C^*\}$ to each branch $b \in E_T$ is* valid *if the two following conditions hold:*

- *(Phylogenetic correctness condition) There must be at most one branch b such that $s(b) = I$.*
- *(Star condition) Let $b \in E_T$, and let $anc(b) \subset E_T$ be the set of branches on the path from the root to b. Then $s(b) \in \{I^*, D^*, C^*\}$ if and only if $\exists b' \in anc(b)$ such that $s(b') = D$ or $\exists b' \in (E_T \setminus anc(b))$ such that $s(b') = I$.*

The number of valid states on a complete balanced phylogenetic tree with n leaves is $O(n \cdot 3^{2n})$ (the number is dominated by states that have a 'I' on a branch leading to a leaf, which leaves all other $2n-3$ edges free to be labeled with either C^*, D^*, or I^*). Although this number remains exponential, it is significantly better than the 6^{2n-2} valid and invalid states.

3.2 Emission Probabilities

In an HMM, each state emits one symbol, according a certain emission probability distribution. In our tree-HMMs, each state emits a collection of symbols, corresponding to the set of characters obtained at the leaves of T when indel scenario s occurs. Intuitively, we can think of a state as emitting an alignment column. The following definition formalizes this.

Definition 6. *Let s be a valid state for tree $T = (V_T, E_T)$ with root r. Then, we define the output of state s as a function $O_s : V_T \to \{0, 1\}$ with the following recursive properties:*

1. $O_s(root) = \begin{cases} 0, & \text{if } \exists x \in V_T \text{ such that } s(x) = I \\ 1, & \text{otherwise} \end{cases}$.
2. *Let $e = (x, y) \in E_T$, with x being the parent of y. Then,*

$$O_s(y) = \begin{cases} 0, & \text{if } s(e) = D \\ 1, & \text{if } s(e) = I \\ O_s(x), & \text{otherwise} \end{cases}$$

Let C be an alignment column (i.e. an assignment of 0 or 1 to each leaf in T). We then have the following degenerate emission probability for state s:

$$Pr_e[C|s] = \begin{cases} 1 \text{ if } O_s(x) = C(x) \text{ for all } x \in leaves(T) \\ 0 \text{ otherwise} \end{cases}$$

Thus, each state s can emit a single alignment column C. However, many different states can emit the same column.

3.3 Transition Probabilities

The last component to be defined is the set of transition probabilities of the tree-HMM. The probability of transition from state s to state s', $\Pr_t[s'|s]$ is a function of set of events that occurred along each edge of T. Intuitively, $\Pr_t[s'|s]$ describes the probability of the single-column indel scenario s', given that scenario s occurred at the previous column. This transition probability is a function of insertions and deletions that started between the two columns, of those that were extended going from one column to the next. Specifically, we have $\Pr_t[s'|s] = \prod_{b \in E_T} \rho[s'(e)|s(e), b]$, where ρ is given in Table 1.

Table 1. Edges transition table $\rho[s'(e)|s(e), b]$. Notice that ρ is not a transition probability matrix, since its rows sum to more than one.

$s(e) \setminus s(e)'$	C	D	I	C^*	D^*	I^*
C	$P_{Cons}(\lambda(b))$	$P_{DelStart}(\lambda(b))$	$P_{InsStart}(\lambda(b))$	1	0	0
D	$(1-\alpha_D)P_{Cons}(\lambda(b))$	α_D	$(1-\alpha_D)P_{InsStart}(\lambda(b))$	0	1	0
I	$(1-\alpha_I)P_{Cons}(\lambda(b))$	$(1-\alpha_I)P_{DelStart}(\lambda(b))$	α_I	0	0	1
C^*	$P_{Cons}(\lambda(b))$	$P_{DelStart}(\lambda(b))$	$P_{InsStart}(\lambda(b))$	1	0	0
D^*	$(1-\alpha_D)P_{Cons}(\lambda(b))$	α_D	$(1-\alpha_D)P_{InsStart}(\lambda(b))$	0	1	0
I^*	$(1-\alpha_I)P_{Cons}(\lambda(b))$	$(1-\alpha_I)P_{DelStart}(\lambda(b))$	α_I	0	0	1

3.4 Tree-HMM Paths and Ancestral Reconstruction

We now show how the tree-HMM described above allows us to solve the IMLP. Consider a multiple alignment A of length L on a tree T. A path π in the tree-HMM is a sequence of states $\pi = \pi_0, \pi_1, ..., \pi_L, \pi_{L+1}$. Based on the standard HMM theory, we get:

$$Pr[\pi, A] = Pr[\pi_0, A_0] \prod_{i=1}^{L+1} Pr_e[A[i]|\pi_i] \cdot Pr_t[\pi_i|\pi_{i-1}]$$

Figure 3 gives an example of an alignment with some of the non-zero probability paths associated.

Theorem 1. *Consider an alignment A on a tree T. Then $\pi^* = argmax_\pi \Pr[\pi, A]$ yields the most likely indel scenario for A, and a maximum likelihood ancestral reconstruction A^* is obtained by setting $A^*_u[i] = O_{\pi^*_i}(u)$.*

Proof. It is simple to show that for any ancestral reconstruction $\hat{A}$ for A, we have $L(\hat{A}) = \Pr[\pi, A]$, where π is the path corresponding to $\hat{A}$. Thus, maximizing $\Pr[\pi, A]$ maximizes $L(\hat{A})$.

4 Computing the Most Likely Path

To compute the most likely path π^* through a tree-HMM, we adapted the standard Viterbi dynamic programming algorithm [6]. Let

$$X(i,k) = \max_{\substack{\pi = \pi_0, \pi_1, ..., \pi_i \\ \text{such that } \pi_i = k}} \Pr[\pi, A[1...i]]$$

be the likelihood of the most probable path ending at state k for the i first columns of the alignment. Let $c \in \mathcal{S}$ be the state made of C's on all edges of T. Since the dummy column $A[0]$ consists exclusively of 1's, c is the only possible initial state. For any i between 0 and $L+1$ and for any valid state $s \in \mathcal{S}$, we can compute $X(i,s)$ as follows:

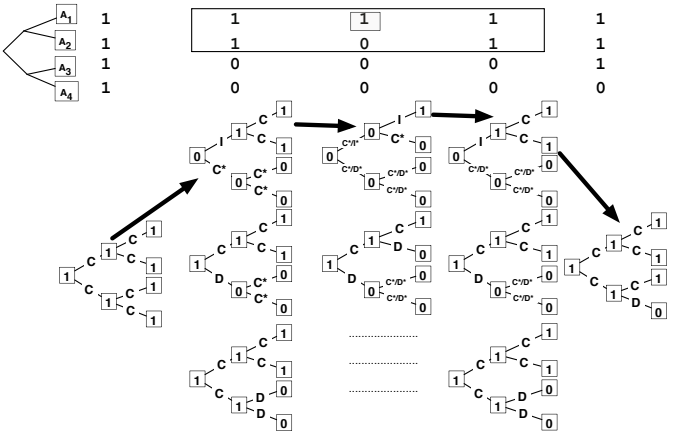


Fig. 3. The set of valid, non-zero probability states associated to the multiple alignment given at the top of the figure. When edges are labeled with more than one character (e.g. C^*, D^*), the tree represents several possible states. For the third column, not all possible states are shown. Arrows indicate one possible path through the tree-HMM. This path corresponds to two interleaved insertions, shown by two boxes in the alignment, illustrating the need for the I^* character.

$$X(i,s) = \begin{cases} 1, & \text{if } i = 0 \text{ and } s = c \\ 0, & \text{if } i = 0 \text{ and } s \neq c \\ Pr_e[A[i]|s] \cdot \max_{s' \in \mathcal{S}}(X(i-1,s') \cdot Pr_t[s|s']), & \text{if } i > 0 \end{cases}$$

Finally, π^* is obtained by tracing back the dynamic programming, starting from entry $X(L+1, c)$. The running time of a naive implementation of the Viterbi algorithm is $O(|\mathcal{S}|^2 L)$, which quickly becomes impractical as the size of the tree T grows. In the next section, we show how to make this computation practical for moderately large trees and for long sequences.

4.1 Viterbi Optimizations

The previous implementation of the Viterbi algorithm cannot run on large sequences or when the number of taxa is greater to 8, due to the fact that the number of possible valid states is $O(n3^{2n})$. Even though the number of states is exponential, most of alignment columns can only be generated with non-zero probability by a much more manageable number of states. Given an alignment A, it is possible to compute, for each column $A[i]$, the set S_i of valid states that can emit $A[i]$ with non-zero probability. For instance, an alignment column with only 1's will lead to only one possible state, independently of the number taxa n. To compute only the valid states for a given column of the alignment, we used a divide and conquer approach that is presented in Algorithm 1. The idea behind this algorithm is to compute partial valid states for subtrees and to merge these subtrees while keeping only valid merged states. The process is done in a bottom up fashion until the root of the tree is reached.

Although the sets of possible states $S_0, ..., S_{L+1}$ obtained from this algorithm are generally relatively small, more improvements are possible, because the transition probability between most pairs of states is zero. We can thus remove from S_i any state s such that the transition to s from any state in S_{i-1} has probability zero. Proceeding from left to right, we get $S'_0 = S_0$, and $S'_i = \{s \in S_i | \exists t \in S'_{i-1} \text{ s.t. } Pr_t[s|t] > 0\}$. For instance, if, in all states of S_{i-1}, an edge e is labeled by deletion D, then none of the states in S_i can have edge e labeled with C^* or I^*. This yields a huge improvement for alignment regions consisting of a number of adjacent positions with a base in only one of the n species and ensures that the algorithm will be practical for reasonable number of sequences (see Section 5).

Algorithm 1. buildValidState(node *root*, C)

Require: *root*: a tree node, C: an alignment column.
Ensure: Set of valid, non-zero probability states for C.
1: **if** *root* is a leaf **then**
2: **return** list of possible operations according to the character at that leaf
3: **else**
4: $leftList$ = buildValidState(*root*.left, C)
5: $rightList$ = buildValidState(*root*.right, C)
6: **return** $mergeSubtrees(leftList, rightList, root)$
7: **end if**

4.2 Forward-Backward Algorithm

A significant advantage of the maximum likelihood approach over the parsimony approach is that it allows evaluating the uncertainty related to certain aspects of the reconstruction. For example, it is useful to be able to compute the probability that a base was present at a given position i of a given ancestral node u: $\Pr[A^*_u[i] = 1|A] = \sum_{s \in \mathcal{S}: O_s(u)=1} \Pr[\pi_i = s|A]$. This allows the computation of the probability of making an incorrect prediction at a given position of a given ancestor. The forward-backward is a standard HMM algorithm to compute $\Pr[\pi_i = s|A]$ (see [6] for details):

$$F(i,s) = \begin{cases} 1, & \text{if } i = 0 \text{ and } s = c \\ 0, & \text{if } i = 0 \text{ and } s \neq c \\ Pr_e[A[i]|s] \cdot \sum_{s' \in \mathcal{S}'_{i-1}} (F(i-1,s') \cdot Pr_t[s|s']), & \text{if } i > 0 \end{cases}$$

$$B(i,l) = \begin{cases} 1, & \text{if } i = L+1 \text{ and } l = c \\ 0, & \text{if } i = L+1 \text{ and } l \neq c \\ \sum\limits_{s' \in \mathcal{S}'_{i+1}} Pr_e[A[i+1]|s'] \cdot F(i+1,s') \cdot Pr_t[s'|s]), & \text{if } i < L+1 \end{cases}$$

$$\Pr[\pi_i = s|A] = \frac{F(i,s)B(i,s)}{\sum_{s' \in S'_i} F(i,s')B(i,s')}$$

The optimizations developed for the Viterbi algorithm can be also applied directly to the Forward-Backward algorithm.

Algorithm 2. mergeSubtrees(StateList $leftList$, StateList $rightList$, node $root$)

Require: $leftList$ and $rightList$: the lists of partial states, $root$: a tree node.
Ensure: Set of valid, non-zero probability states combining elements in $leftList$ and $rightList$.

```
1: for all partial states l in leftList do
2:   for all partial states r in rightList do
3:     if compatible(l, r) = true then { # merging those two partial states yields a
       valid partial state}
4:       m = merge(l, r)
5:       if root = initialroot then
6:         mergedList.add(m)
7:       else
8:         for all operations on a branch op do
9:           if isPossibleUpstream(m,op) then {#Checks if op can legally be
             added to m}
10:            mergedList.add(addAncestorBranch(m,op))
11:          end if
12:        end for
13:      end if
14:    end if
15:  end for
16: end for
17: return mergedList
```

5 Results

Our tree-HMM algorithm was implemented as a C program that is available upon request. The program was applied to a 50kb region of chromosome 13 of human, together with orthologous regions in 7 other species of mammals: chimp, mouse, rat, cow, dog, armadillo, and elephant[1] [14]. This region is representative of the whole genome, and contains coding, intergenic regions, and intronic regions. The multiple alignment of these regions, computed using TBA [2], contains 85,000 columns. The phylogenetic tree used for the alignment and for the reconstruction is shown in Figure 4. The branch lengths are based on rates of substitution estimated on a genome-wide basis. The parameters of the indel model were set as follows: $\psi_D = 0.05, \psi_I = 0.05, \alpha_D = 0.9$ and $\alpha_I = 0.9$.

We first compared the maximum likelihood ancestral reconstruction found using our Viterbi algorithm to the ancestors inferred using the greedy algorithm of Blanchette *et al.* [1]. Table 2 shows the degree of agreement between the two reconstructed ancestors, for each ancestral node. We observe that both methods agree to a very large degree. The most disagreement concerns the ancestor at the root of the eutherian tree, which, in the absence of an outgroup, cannot be reliably predicted by any method. We expect that in most cases of disagreement,

[1] In the case of cow, armadillo, and elephant, the sequence is incomplete and a small fraction of the bases are missing.

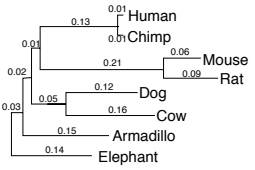


Fig. 4. Phylogenetic tree for the eight mammalian species studied in this paper

Table 2. Percentage of alignment columns where the ancestor reconstructed by the greedy algorithm of Blanchette *et al.* [1] and that predicted by our maximum-likelihood algorithm are in agreement.

Ancestor	**% of agreement**
Cow + Dog	99.38
Mouse + Rat	97.83
Human + Chimp	99.83
Human + Chimp + Mouse + Rat	99.33
Human + Chimp + Mouse + Rat + Cow + Dog	98.41
Human + Chimp + Mouse + Rat + Cow + Dog + Armadillo	94.13
Human + Chimp + Mouse + Rat + Cow + Dog + Armadillo + Elephant	89.01

the maximum likelihood is the most likely to be correct, although the opposite may be true in case of gross model violations [11].

The main strength of the likelihood-based method is its ability to measure uncertainty, using the forward-backward algorithm, which something that no previous method allowed. The probability that the maximum posterior probability reconstruction is correct is simply given by $\max\{\Pr[A_u^*[i] = 1|A], 1 - \Pr[A_u^*[i] = 1|A]\}$. For example, if $\Pr[A_u^*[i] = 1|A] = 0.3$, then the maximum posterior probability reconstruction would predict $A_u^*[i] = 0$, and would be right with probability 0.7. Figure 5 shows the distribution of this probability of correctness, for each ancestral node in the tree. We observe, for example, that 97.7% of the positions in the Boreoeutherian ancestor (the human+chimp+mouse+rat+cow+dog ancestor, living approximately 70 million years ago), are reconstructed with a confidence level above 99% [2]. The ancestor that is the easiest to reconstruct confidently is obvious the human-chimp ancestor, where less than 0.5% of the columns have a confidence level below 99%. Again, the root of the tree is the node that is the most difficult to be reconstructed confidently, because of the absence of an outgroup. Overall, this shows that most positions of most ancestral nodes can be reconstructed very accurately, and that we can identify the few positions where the reconstruction is uncertain.

A potential drawback of the tree-HMM method is that it's running time is, in the worst case, exponential on the number of sequences being compared. However, the optimizations described in this paper greatly reduce this number, so

[2] We need to to keep in mind, though, that these numbers assume the correctness of the multiple alignment, as well as that of the branch lengths and indel probability model, so that they do not reflect the true correctness of the reconstructed ancestor.

the algorithm remains quite fast. Our optimized Viterbi algorithm produced its maximum likelihood ancestral predictions on the 8-species alignment of 85,000 columns in two hours on an Intel Pentium IV machine (3.2 Ghz), while the forward-backward algorithm produced an output after approximately four hours. Figure 6 shows the distribution of the number of states that were actually considered, per alignment column, in the case of the 8-species alignment of 85,000 columns. Most alignment column are actually associated to less than 50 states. However, a small number of columns are associated to a very large number of states (8 columns have more than 21,000 states). Fortunately, these columns are rarely consecutive, so the incurred running time is not catastrophic.

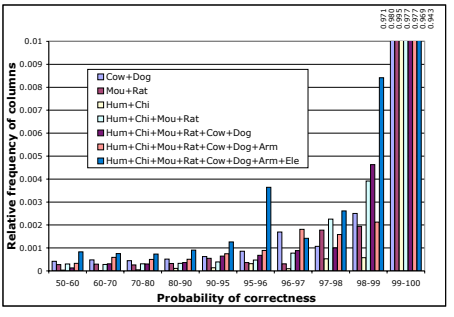


Fig. 5. Distribution of the confidence levels, over all 85,000 columns, for each ancestor. The vast majority of the ancestral positions are reconstructed with a probability of correctness above 99% (assuming the correctness of the alignment).

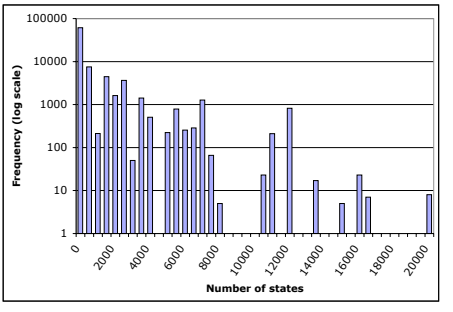


Fig. 6. Distribution of the number of states considered ($|S'_i|$), over all 85,000 positions

6 Discussion and Future Work

The method developed here allows predicting maximum likelihood indel scenarios and their resulting ancestral sequences for reasonably large alignments. Furthermore, it allows estimating of the probability of error in any part of the prediction, using the forward-backward algorithm. Integrated into the pipeline for whole-genome ancestral reconstruction, it will improve the quality of the predictions and allow richer analyses. The main weakness of our approach is that it assumes that a correct phylogenetic alignment is given as input. While many existing multiple alignment programs have been shown to be quite accurate on mammalian genomic sequences (including non-functional or repetitive regions) [2], it has been also shown that a sizeable fraction of reconstruction errors is due to incorrect alignments [1]. Ideally, one would include the optimization of the alignment directly into the indel reconstruction problem, as originally suggested by Hein [10]. However, with the exception of statistical alignment approaches [13] (which remains too slow to be applicable on a genome-wide scale), genomic

multiple alignment methods do not treat indels in a probabilistic framework. We are thus investigating the possibility of using the method proposed here to detect certain types of small-scale alignment errors, and to suggest corrections.

When predicting ancestral genomic sequences, it is very important to be able to quantify the uncertainty with respect to certain aspects of the reconstruction. Our forward-backward algorithm calculates this probability of error for each position of each ancestral species. However, errors in adjacent columns are not independent: if position i is incorrectly reconstructed, it is very likely that position $i+1$ will be wrong too. We are currently working on models to represent this type of correlated uncertainties. This new type of representation will play an important role in the analysis and visualization of ancestral reconstructions.

Finally, to be applicable to complete genomes, and to scale up to the $\sim$20 mammalian genomes that will soon be available, our algorithm will require further optimizations. These will probably force us to move away from an exact algorithm toward approximation algorithms.

Acknowledgements

A.B.D. is an NSERC fellow. We thank Éric Gaul, Éric Blais, Adam Siepel, and the group of participants to the First Barbados Workshop on Paleogenomics for their useful comments. We thank Webb Miller and David Haussler for providing us with the sequence alignment data.

References

1. M. Blanchette, E. D. Green, W. Miller, and D. Haussler. Reconstructing large regions of an ancestral mammalian genome in silico. *Genome Res*, 14(12):2412–2423, Dec 2004.
2. M. Blanchette, W. J. Kent, C. Riemer, L. Elnitski, A. F. A. Smit, K. M. Roskin, R. Baertsch, K. Rosenbloom, H. Clawson, E. D. Green, D. Haussler, and W. Miller. Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Research*, 14(4):708–715, Apr 2004.
3. N. Bray and L. Pachter. MAVID: constrained ancestral alignment of multiple sequences. *Genome Research*, 14(4):693–699, Apr 2004.
4. M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, and S. Batzoglou. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Research*, 13(4):721–731, Apr 2003.
5. L. Chindelevitch, Z. Li, E. Blais, and M. Blanchette. On the inference of parsimonious indel evolutionary scenarios. *Journal of Bioinformatics and Computational Biology*, 0:In press, 2006.
6. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.
7. J. Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.
8. J. Felsenstein and G. Churchill. A hidden markov model approach to variation among sites in rate of evolution. *Mol. Biol. Evol.*, 13:93–104, 1996.

9. J. Fredslund, J. Hein, and T. Scharling. A large version of the small parsimony problem. In *Proceedings of the 4th Workshop on Algorithms in Bioinformatics (WABI)*, 2004.
10. J. Hein. A method that simultaneously aligns, finds the phylogeny and reconstructs ancestral sequences for any number of ancestral sequences. *Molecular Biology and Evolution*, 6(6):649–668, 1989.
11. A. Hudek and D. Brown. Ancestral sequence alignment under optimal conditions. *BMC Bioinformatics*, 6:273:1–14, 2005.
12. W. J. Kent, R. Baertsch, A. Hinrichs, W. Miller, and D. Haussler. Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proc Natl Acad Sci U S A*, 100(20):11484–11489, Sep 2003.
13. G. Lunter, I. Miklos, Y. Song, and J. Hein. An efficient algorithm for statistical multiple alignment on arbitrary phylogenetic trees. *J Computational Biology*, 10(6):869–89, 2003.
14. W. Miller. Personal communication.
15. E. Rivas. Evolutionary models for insertions and deletions in a probabilistic modeling framework. *BMC Bioinformatics*, 6(1):63, 2005.
16. A. Siepel, G. Bejerano, J. S. Pedersen, A. S. Hinrichs, M. Hou, K. Rosenbloom, H. Clawson, J. Spieth, L. W. Hillier, S. Richards, G. M. Weinstock, R. K. Wilson, R. A. Gibbs, W. J. Kent, W. Miller, and D. Haussler. Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Res*, 15(8):1034–1050, Aug 2005.
17. A. Siepel and D. Haussler. Combining phylogenetic and hidden markov models in biosequence analysis. *J Comput Biology*, 11(2-3):413–28, 2004.
18. J. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: an improved likelihood model of sequence evolution. *J. Mol. Evol.*, 34:3–16, 1992.
19. J. L. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences. *J Mol Evol*, 33(2):114–124, Aug 1991.
20. Z. Yang. Among-site rate variation and its impact on phylogenetic analysis, 1996.

Conservation Patterns in *cis*-Elements Reveal Compensatory Mutations

Perry Evans, Greg Donahue, and Sridhar Hannenhalli[1,*]

Genomics and Computational Biology
[1] Department of Genetics
University of Pennsylvania
Philadelphia, PA 19104-6021
evansjp@mail.med.upenn.edu, gdonahue@mail.med.upenn.edu
sridharh@pcbi.upenn.edu

Abstract. Transcriptional regulation critically depends on proper interactions between transcription factors (TF) and their cognate DNA binding sites or *cis* elements. A better understanding and modelling of the TF-DNA interaction is an important area of research. The Positional Weight Matrix (PWM) is the most common model of TF-DNA binding and it presumes that the nucleotide preferences at individual positions within the binding site are independent. However, studies have shown that this independence assumption does not always hold. If the nucleotide preference at one position depends on the nucleotide at another position, a chance mutation at one position should exert selection pressures at the other position. By comparing the patterns of evolutionary conservation at individual positions within *cis* elements, here we show that positional dependence within binding sites is highly prevalent. We also show that dependent positions are more likely to be functional, as evidenced by a higher information content and higher conservation. We discuss two examples—Elk-1 and SAP-1 where the inferred compensatory mutation is consistent with known TF-DNA crystal structure.

1 Introduction

Gene transcription in eukaryotes is regulated by a network of transcription factors (TF) [1, 2]. A first step in analyzing transcription is to model each TF's DNA binding preference. The *in vitro* SELEX [3] or footprinting [4] experiments can be used to derive the TF's binding specificity as a *positional weight matrix (PWM)* [5]. Although PWMs are used as the *de facto* model of TF-DNA interaction, they assume that the nucleotide preferences at individual positions within binding sites are independent. However, there are several direct and indirect evidences that support inter-position dependence within binding sites. Dependence among positions was experimentally shown for Mnt repressor [6]. The binding site specificity model can be improved by incorporating local pairwise nucleotide dependence [7]. Because of this inter-position dependence, the binding sites may fall into distinct clusters or subclasses. This

* Corresponding author.

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 186 – 199, 2006.

hypothesis was exploited to improve binding site modeling *via* a mixture model of PWMs [8].

In a biological system with interacting components, mutation in one component will lead to selection pressure on the interacting components. Compensatory changes and co-evolution of functionally interacting components are not uncommon [9-12]. Given several examples of experimentally determined binding sites for a factor, one can assess positional dependence by computing the correlation among nucleotide pairs at two positions. For instance, the occurrence of nucleotide pair *x,y* at position-pair *i,j* at a much higher frequency than expected, indicates dependence between positions *i*, and *j*. This and related ideas have been used previously in different contexts of DNA signals [13, 14]. However, if we think of the TF-DNA complex as a system and the individual nucleotides as components, the positional dependence can be assessed more directly by comparing the evolutionary histories of the two positions. Specifically, if a chance mutation at position *i* increases the selection pressure for a compensatory mutation at position *j*, this mutation pattern should appear in the phylogeny of the binding site, and its presence indicates a functional dependence of position *j* on position *i*.

Here we present a method to assess compensatory mutations within TF binding sites. We use maximum parsimony tree(s) to infer the evolutionary history of each position of the binding site. We then compare the conditional probability of a tree (conditioned upon the tree at another position) to its unconditional probability. A higher conditional probability indicates compensatory mutation. By using whole genome multiple alignments of 5 mammalian genomes and their phylogenetic relationship, and high confidence putative binding sites in human promoters for the vertebrate transcription factor PWMs from JASPAR [15], we show that positional dependence within binding sites is highly prevalent, especially at adjacent positions. We also show that inter-positional dependencies are more likely to be functional, as evidenced by a greater information content and conservation. We further discuss a few cases that could be corroborated by the literature. Thus this work, for the first time, directly exploits the pattern of evolutionary changes in individual binding site positions to assess compensatory mutations.

2 Results

Putative binding sites in human promoters. To estimate compensatory mutations one needs a large collection of binding site instances - ideally, experimentally determined. Because of a lack of experimentally determined binding sites, we used our previously reported computational approach [16] based on phylogenetic footprinting. For each of the 79 vertebrate PWMs in JASPAR [15], we scanned the 1kb promoter region of 20835 human RefSeq genes using our PWM_SCAN tool [16] with a p-value threshold of e^{-9} (chance expectation of one hit every 8.1kb of human genome). Among these initial matches we retained up to 1000 highest scoring matches that fell within a gapless multiple alignment of human-chimp-mouse-rat-dog (genome.ucsc.edu). Fifteen PWMs had no match qualifying these two criteria, thus our analysis is based on 64 PWMs. The numbers of matches per TF varied from 1000 to 139 with an average of 856.

Compensatory mutations between binding site positions. For a transcription factor M, let B^M be the set of matches (putative binding sites) in promoters; $|B^M| = N^M$. Let L^M be the width of the binding site. We assume the widely accepted evolutionary relationship among the 5 mammals. At each position of each of the sites, we inferred the ancestral nucleotides from the maximum parsimony tree(s) using Fitch's algorithm [17]. All maximum parsimony trees were used in the calculations and were assumed to be equiprobable. Let *CompMut(M,u,v)* be the compensatory mutation for factor M between positions u and v (see Methods). High values of *CompMut(M,u,v)* indicate compensatory mutations. We define scope $s=v-u$, assuming $v > u$ [7]. *CompMut(M,u,v)* is computed based on the u,v position pairs of all matches of M (see above). To test the significance of *CompMut(M,u,v)*, we generated a control set of i,j position pairs. We have used 4 different controls with increasing stringency. For all controls we only consider the gapless aligned positions in the 1kb promoter regions of 20835 human genes (see above). The 4 controls are:

1. Control-1 constructs i, j pairs by randomly choosing positions i and j. Thus, for a set of N^M positive position pairs, a set of N^M is randomly selected and treated like the positive pairs.
2. Control-2 constructs i,j pairs by randomly choosing position i and then selecting $j=i+s$. The only difference from Control-1 is that now the position j is not independent from position i. This control captures the inherent neighbourhood dependence.
3. Control-3 constructs PWM M^r with same width as M by randomly sampling columns from the 79 vertebrate PWMs in JASPAR. We identify all matches of M^r following an identical procedure as for M and select i,j position pairs from these matches.
4. Control-4 derives PWM M^r from M by randomly shuffling the compositions at each column (position). Then we identify all matches of M^r following an identical procedure as for M and select u,v position pairs from these matches.

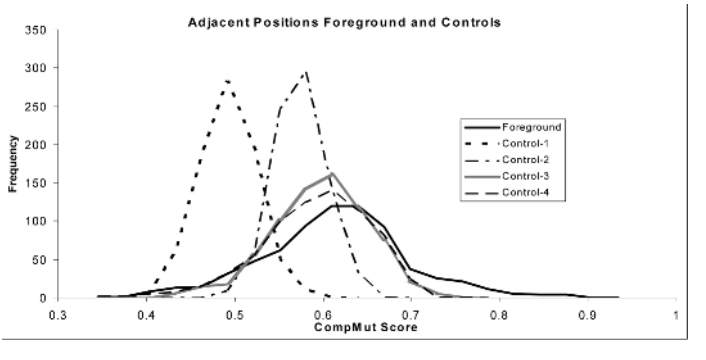


Fig. 1. At *scope*=1, the *CompMut* values are significantly higher for the actual binding sites relative to all 4 controls with a Wilcoxon rank rum based p-value of 5E-8, relative to Control-4

Control-1 is the least stringent control and represents random expectation of compensatory mutations, while Control-2 accounts for local dependencies in the genomes, which may in part correspond to cis elements undetected by our initial prediction or to other functional elements. Control-3,4 are the most stringent controls in that they recreate PWMs which bear the characteristics of real PWMs. Fig. 1 shows the distributions of all *CompMut* values for 64 JASPAR transcription factors at scope *s=1*. The figure shows the plot for the actual binding sites (foreground) against the 4 controls. Even against the most stringent, Control-4, the foreground has significantly higher values of *CompMut*. The Wilcoxon rank sum test based p-value against Control-4 is 4.9×10^{-8}. Similar relationships hold for larger scopes (data not shown). However, as we will report next, for higher scopes there is a monotonic decrease in compensatory mutations.

Compensatory mutation decreases with increasing distance between positions. We repeated the above experiment for all scopes. With increasing scope, the compensatory mutations become smaller and less differentiable from the controls (Fig. 2). Thus the inter-position dependence seems to be more prevalent for nearby positions, especially for adjacent positions. The Wilcoxon rank-sum test p-value against Control-4 varies with increasing scope as follows: **1**: 4.9×10^{-8}, **2**: 5.6×10^{-6}, **3**: 5.6×10^{-6}, **4**: 4.7×10^{-7}, **5**: 8.8×10^{-4}, **6**: 6.1×10^{-3}, **7**: 0.02, **8**: 0.09.

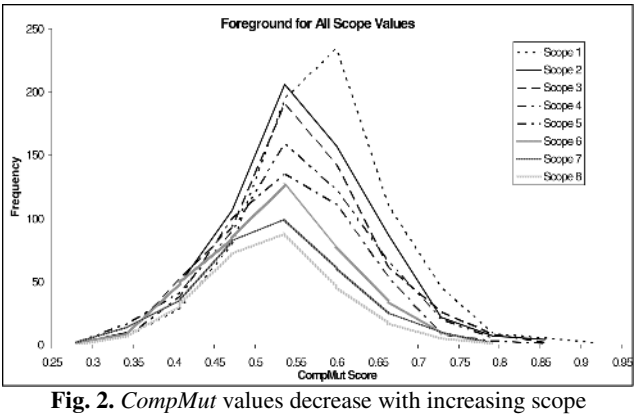


Fig. 2. *CompMut* values decrease with increasing scope

Functional relevance of positions with compensatory mutations. We expect that the positions with greater compensatory mutations are under selection pressure and thus more likely to be functional. Due to lack of experimental data to verify whether an individual position is functional, we have employed an indirect approach. Functionally relevant positions are likely to be less variable (higher information content [18] in the PWM) and also likely to have greater conservation across species; the phylogenetic footprinting-based prediction of functional elements is based on this principle. Previous analysis of TF-DNA 3D-structures reveals that the nucleotides interacting with the evolutionarily conserved amino acids have greater information content [12].

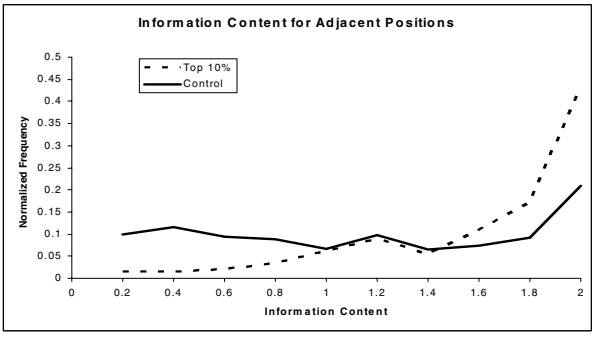


Fig. 3. The positions involved in compensatory mutations have higher information content

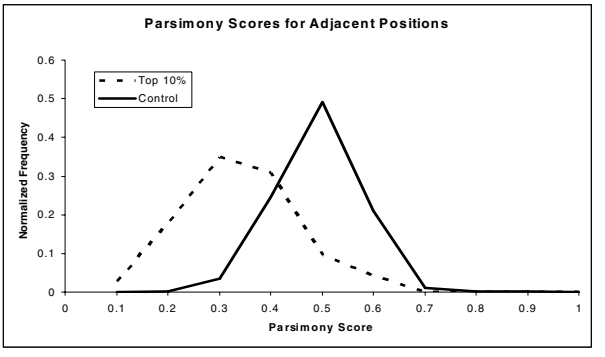


Fig. 4. The positions involved in compensatory mutations are evolutionarily more conserved

Note that there is no favourable *a priori* bias in our measures for compensatory mutation to favour either the information content or the evolutionary conservation. To an extreme case, for a PWM position with high information content whose occurrences are highly conserved (low parsimony scores), the unconditional tree probabilities will be very high and thus we will nt detect a compensatory mutation involving this position. We selected the top 10% of the adjacent position pairs (*scope*=1) with highest *CompMut* values. We used all positions that were part of these pairs as the positive set. The remaining positions were used as the control. We compared the information content and the conservation for the positive and the control set. We used the average parsimony score as a proxy for conservation—the lower the parsimony score, the higher the conservation. Fig. 3 shows that the positions involved in compensatory mutations have a higher information content (Wilcoxon p-value = 0). A similar conclusion can be reached for evolutionary conservation as shown in Fig. 4 (Wilcoxon p-value = 0).

Prediction and validation of specific compensatory mutations. So far we have shown that the adjacent positions in putative binding sites show compensatory mutations and the positions that exhibit compensatory mutations have greater information content

and are more conserved. Applying a stringent threshold for *CompMut* allows us to predict specific position pairs with compensatory mutations. We have used the 99th percentile of the Control-4 as the threshold for *CompMut*. A total of 280 pairs for all scopes qualify this threshold. Supplementary Table S1 provides the pairs of positions separately for each scope (only top 20 pairs are shown for each scope). These results can be best interpreted against 3-D structures of protein-DNA interactions, but this is currently possible only for a handful of cases. Here, we examine a few cases for which literature data is available for direct verification.

ETS Family. JASPAR has 5 vertebrate PWMs in the ETS family, which is characterized by a GGA core. Due to the significance threshold on PWM hits, only three of these were analyzed: MA0028, MA0062, and MA0076. All three have significant *CompMut* scores involving the core positions. A detailed protein-DNA interaction diagram showing amino acid/nucleotide interactions is available for MA0028 (Elk-1) [19]. Fig. 5 shows that for MA0028, individual amino acids interact with multiple bases, suggesting dependencies between bases. Fig. 5 also shows all predicted position pairs with compensatory mutations that qualify the 99th percentile threshold relative to the respective Control-4 values of *CompMut*. The predicted pairs are clustered around the GGA core. Furthermore, all of the pairs interacting with the same amino acid are predicted (shown as arches), and also a few that do not interact with the same amino acid (dotted arches).

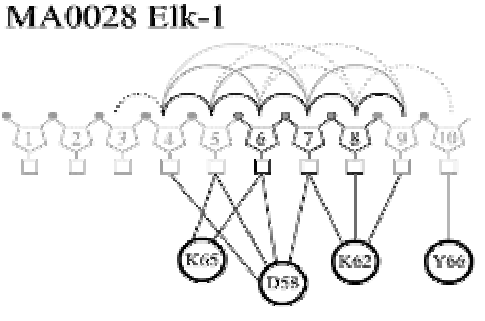


Fig. 5. The MA0028 PWM is represented as a chain of 10 bases, with the core in darker font. Amino acid interactions are indicated below the chain. Significant *CompMut* pairs are indicated by arches above the chain. *CompMut* pairs denoted by dotted lines are not joined by an amino acid interaction. Different scopes are distinguished by the darkness of the arch.

There is also TF-DNA structural information available for MA0076 (SAP-1) [20]. Fig. 6 shows residues of SAP-1 protein α3 helix interacting with DNA bases. As before, all the nucleotide pairs contacting the same residue have high value of *CompMut* (99% pf the Control-4). However, we infer several other dependencies that are not directly supported by shared residue contacts. Nevertheless, these additional inferred dependencies are supported by shared contact with the α3 helix, *i.e.*, positions 3 through 8. Furthermore, it is suspected that the base at position 2 influences the stability of the bond between the alpha helix and the recognition site, so position 2 should also be added to the interdependent set of positions. Thus there are

21 position pairs and all of these qualify the 99th percentile threshold relative to the respective Control-4 values of *CompMut*. Five additional significant dependencies are detected with the *CompMut* algorithm, as indicated in Fig. 6, and each of these has one base that interacts with the alpha helix.

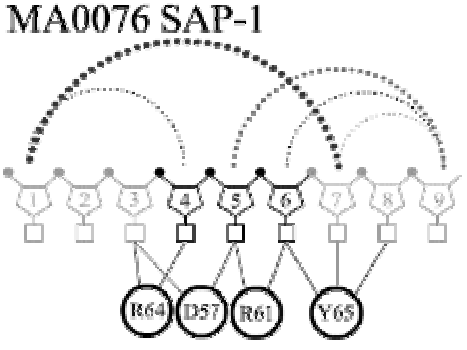


Fig. 6. The MA0076 PWM is represented as a chain of 9 bases, with the core in darker font. Amino acid interactions are indicated below the chain. Only the significant *CompMut* pairs that involve a non helix residue are shown by dotted lines. Different scopes are distinguished by the darkness of the arch. Significant *CompMut* pairs within the alpha helix are not shown to simplify the figure.

SAP-1 binds to he *Drosophila* E74 promoter and the *Human* c-*fos* promoter [20]. SAP-1 has a higher affinity for the fly promoter than for the human promoter, but the two binding sites differ only at positions 3 and 7 in Fig.6. In the fly, a C at position 3 causes the DNA to adopt characteristics that favour the binding of the alpha helix mentioned above. When the C is changed to an A in the *Human* promoter, these characteristics are lost and the binding is less favourable. 3, 7 is a significant *CompMut* pair. The observations are consistent with a change at position 7 in the *Human* promoter to compensate for the loss of affinity (by changing from an A in *Drosophila* to a T in *Human*). Relative importance of bases T and A at position 7 is however not known.

3 Methods

Computing Compensatory Mutation. We compute the probability of a tree using the Markovian process as in PhyME [21]. However, we differentiate between the unconditional probability and the probability conditioned on the other position. The compensatory mutation score for positions *i* and *j* for PWM *X* is the fraction of the PWM matches of X for which the probability of seeing a tree in one position, *i* or *j,* is increased by knowledge about the trees at the other position. To calculate compensatory mutations, we used two tree probability models, one unconditional and one conditioned on a tree from the other position in the *i*,*j* pair. We represent these

two models using recursive equations on the tree. Below we use the same notation to denote a tree node and the nucleotide at that node.

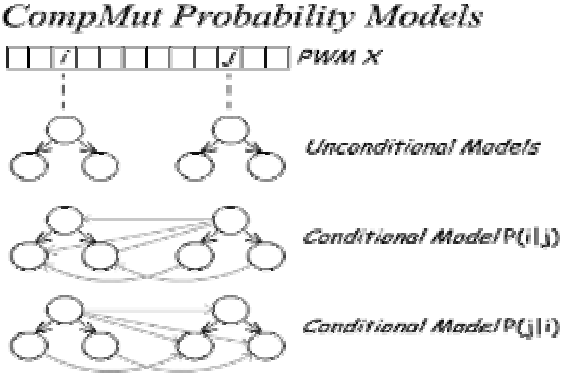


Fig. 7. We consider trees from positions *i* and *j* in PWM hits and compare the unconditional and conditional probabilities of these trees. The *CompMut* is the percentage of trees for which the conditional probability is higher than the unconditional probability.

Unconditional model. The unconditional probability of a tree is computed based on a Markov process where the root of the tree has a specific emission probability for the 4 bases, and the various transitions along each tree edge have specific probabilities. The emission and transition probabilities are estimated from all but one tree. To calculate the probability of a tree, we begin at the root and work our way down the tree recursively.

For a subtree rooted at *t,* let *a* be the parent of *t* and let t_L and t_R be the left and right children of *t*. Let *PT(t)* be the probability of subtree rooted at *t*. We use *PT* to denote tree probability to distinguish it from node probability. $PT(t) = \Pr(t \mid a) \times PT(t_L) \times PT(t_R)$. If *t* is the root of the tree, then $\Pr(t \mid a) = \Pr(t)$, and for leaf nodes, the children probabilities are set to 1.

Conditional model. As in the unconditional model, the probability of a tree is estimated using a Markov process. However, the emission probability at the root is conditional upon the root nucleotide at another position. The transition probability at an edge now depends on the given transition in the corresponding branch at another position. In other words, what is the probability of transitioning from A to a C given that there was a transition from G to an A in the corresponding branch in the other position? As in the unconditional model, the probability calculation begins at the root and moves down the tree recursively, as described below.

For a pair of positions and the trees at the two positions, let t^1 and t^2 be the roots of the subtrees rooted at some common ancestral species at the two positions. Let *a* and *b* be the parent nodes of t^1 and t^2 respectively. Let $PT(t^1 \mid t^2)$ be the probability of t^1 conditioned upon t^2 .

$PT(t^1 \mid t^2) = \Pr(t^1 \mid a,b,t^2) \times PT(t_L^1 \mid t_L^2) \times PT(t_R^1 \mid t_R^2)$. As before, if t^1 and t^2 are the root nodes, then $\Pr(t^1 \mid a,b,t^2) = PT(t^1 \mid t^2)$ and for leaf nodes, the children probabilities are set to 1.

Both models are trained using all parsimonious trees constructed at positions *i* and *j* from the PWM hits, and then the tree probabilities above are computed. When more than one parsimonious tree appears at a position, fractional counts are used. Not all joint counts are observed, so pseudocounts are estimated by randomly choosing two positions as in Control-1 and comparing the trees constructed at these positions. We keep track of the number of times the conditional model estimates a higher tree probability than the unconditional model.

For all PWM matches

Train the conditional and unconditional models, excluding the position i *and* j *trees from the training data.*

cond = 0

// T_i: set of all optimal parsimonious trees at position *i*.
// T_j: set of all optimal parsimonious trees at position *j*.

For all tree pairs $(t_1, t_2), t_1 \in T_i, t_2 \in T_j$ *:*

$$cond = cond + \frac{PT(t_1 \mid t_2)}{\mid T_i \mid \times \mid T_j \mid}$$

uncond = 0

For all trees $t \in T_i$ *:*

$$uncond = uncond + \frac{PT(t)}{\mid T_i \mid}$$

$$S_{i,j} = S_{i,j} + \frac{1}{PWM\_hit\_count} \quad \textit{if cond > uncond, 0 otherwise}$$

$S_{i,j}$ is the fraction of times the conditional model $PT(t_i|t_j)$ is greater than the unconditional model $PT(t_i)$.

$$CompMut(M, i, j) = avg(S_{i,j}, S_{j,i})$$

4 Discussion

We have presented methods to assess compensatory mutations within TF binding sites based on inferred patterns of co-mutations at position pairs. We have used several controls to show that there is a significant prevalence of compensatory mutations within binding sites. Our method may suffer from a lack of sufficient data to estimate conditional probabilities of particular changes along the tree branches, thus affecting the robustness. Our approach of using the fraction of trees for which the conditional probability is higher (as opposed to using mean of median, etc), is meant to increase

the robustness. Another method that we have implemented (not reported here) simply counts the number of correlated changes in corresponding tree branches and uses a log-likelihood measure relative to the expected number of correlated changes. This method suffers from a lack of resolution in that it only captures correlated changes along tree branches, without differentiating among the types of changes. In general, for the top position pairs, the two methods agree (Pearson's correlation of ~0.6). However, a detailed comparative analysis has not been performed.

The majority of compensatory mutations are among adjacent positions; with increasing distance between positions, this tendency decreases. Ultimately, evolution is only inferred and any one most parsimonious tree may not be accurate. To make our measures robust, we have relied on all maximum parsimonious trees. We have also shown that the compensating positions are more likely to be functional as evidenced by lower variance within species (high information content) as well as across species (higher conservation). There are very few examples of experimentally determined TF-DNA 3D structures to directly interpret our specific discoveries. However, we have shown for a few cases that our findings are reasonable when compared to experimental data: two ETS family members. Even though, in general, there are few compensatory mutations at greater distances within the binding sites, those would be very interesting to study in greater detail (Tab. S1). We will closely investigate the predictions at higher scopes in the future. These studies will not only provide specific insights into functional interactions between transcription factors and DNA, they can directly inform the efforts to model the DNA binding specificity by incorporating the inter-position dependence.

Acknowledgement

SH is supported by NIH grant R21 GM078203-01.

References

1. Ptashne, M. A genetic switch, Edn. third. (Cold Spring Harbor Laboratory Press, 2004).
2. Kadonaga, J.T. Regulation of RNA polymerase II transcription by sequence-specific DNA binding factors. *Cell* **116**, 247-257 (2004).
3. Tuerk, C. & Gold, L. Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase. *Science* **249**, 505-510 (1990).
4. Guille, M.J. & Kneale, G.G. Methods for the analysis of DNA-protein interactions. *Mol Biotechnol* **8**, 35-52 (1997).
5. Stormo, G.D. DNA binding sites: representation and discovery. *Bioinformatics* **16**, 16-23 (2000).
6. Man, T.K. & Stormo, G.D. Non-independence of Mnt repressor-operator interaction determined by a new quantitative multiple fluorescence relative affinity (QuMFRA) assay. *Nucleic Acids Res* **29**, 2471-2478 (2001).
7. Osada, R., Zaslavsky, E. & Singh, M. Comparative analysis of methods for representing and searching for transcription factor binding sites. *Bioinformatics* **20**, 3516-3525 (2004).
8. Hannenhalli, S. & Wang, L.S. Enhanced position weight matrices using mixture models. *Bioinformatics* **21 Suppl 1**, i204-i212 (2005).
9. Reynolds, M.G. Compensatory evolution in rifampin-resistant Escherichia coli. *Genetics* **156**, 1471-1481 (2000).

10. McGregor, A.P., Shaw, P.J., Hancock, J.M., Bopp, D., Hediger, M., Wratten, N.S. & Dover, G.A. Rapid restructuring of bicoid-dependent hunchback promoters within and between Dipteran species: implications for molecular coevolution. *Evol Dev* **3**, 397-407 (2001).
11. Pischedda, A. & Chippindale, A. Sex, mutation and fitness: asymmetric costs and routes to recovery through compensatory evolution. *J Evol Biol* **18**, 1115-1122 (2005).
12. Raviscioni, M., Gu, P., Sattar, M., Cooney, A.J. & Lichtarge, O. Correlated evolutionary pressure at interacting transcription factors and DNA response elements can guide the rational engineering of DNA binding specificity. *J Mol Biol* **350**, 402-415 (2005).
13. Agarwal, P. & Bafna, V. Detecting non-adjoining correlations with signals in DNA. *RECOMB* (1998).
14. Burge, C. & Karlin, S. Prediction of complete gene structures in human genomic DNA. *J Mol Biol* **268**, 78-94 (1997).
15. Sandelin, A., Alkema, W., Engstrom, P., Wasserman, W.W. & Lenhard, B. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res* **32**, D91-94 (2004).
16. Levy, S. & Hannenhalli, S. Identification of transcription factor binding sites in the human genome sequence. *Mamm Genome* **13**, 510-514 (2002).
17. Fitch, W.M. Toward Defining the Course of Evolution: Minimum Change for a Specified Tree Topology. *Systematic Zoology* **20**, 406-416 (1971).
18. Schneider, T.D., Stormo, G.D., Gold, L. & Ehrenfeucht, A. Information content of binding sites on nucleotide sequences. *J Mol Biol* **188**, 415-431 (1986).
19. Mo, Y., Vaessen, B., Johnston, K. & Marmorstein, R. Structure of the elk-1-DNA complex reveals how DNA-distal residues affect ETS domain recognition of DNA. *Nat Struct Biol* **7**, 292-297 (2000).
20. Mo, Y., Vaessen, B., Johnston, K. & Marmorstein, R. Structures of SAP-1 bound to DNA targets from the E74 and c-fos promoters: insights into DNA sequence discrimination by Ets proteins. *Mol Cell* **2**, 201-212 (1998).
21. Sinha, S., Blanchette, M. & Tompa, M. PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics* **5**, 170 (2004).

Supplementary Material

Table S1. For each scope, the table lists the position pairs that qualify the 99th percentile threshold relative to the respective Control-4 values of *CompMut*.

Scope 1: 80 Values (Top 20 Shown)

PWM ID	PWM Name	Family	Position 1	Position 2
MA0076	SAP-1	ETS	5	6
MA0076	SAP-1	ETS	4	5
MA0076	SAP-1	ETS	6	7
MA0062	NRF-2	ETS	6	7
MA0062	NRF-2	ETS	5	6
MA0062	NRF-2	ETS	4	5
MA0028	Elk-1	ETS	8	9
MA0028	Elk-1	ETS	7	8
MA0088	Staf	ZN-FINGER_C2H2	7	8
MA0076	SAP-1	ETS	3	4

MA0028	Elk-1	ETS	6	7
MA0052	MEF2	MADS	6	7
MA0076	SAP-1	ETS	2	3
MA0058	Max	bHLH-ZIP	8	9
MA0062	NRF-2	ETS	3	4
MA0058	Max	bHLH-ZIP	4	5
MA0018	CREB	bZIP	9	10
MA0079	SP1	ZN-FINGER_C2H2	6	7
MA0018	CREB	bZIP	6	7
MA0052	MEF2	MADS	5	6

Scope 2: 61 Values (Top 20 Shown)

PWM ID	PWM Name	Family	Position 1	Position 2
MA0076	SAP-1	ETS	5	7
MA0062	NRF-2	ETS	5	7
MA0076	SAP-1	ETS	4	6
MA0028	Elk-1	ETS	7	9
MA0062	NRF-2	ETS	4	6
MA0076	SAP-1	ETS	3	5
MA0028	Elk-1	ETS	6	8
MA0076	SAP-1	ETS	2	4
MA0058	Max	bHLH-ZIP	7	9
MA0018	CREB	bZIP	7	9
MA0028	Elk-1	ETS	5	7
MA0088	Staf	ZN-FINGER_C2H2	7	9
MA0079	SP1	ZN-FINGER_C2H2	7	9
MA0028	Elk-1	ETS	4	6
MA0018	CREB	bZIP	6	8
MA0076	SAP-1	ETS	6	8
MA0062	NRF-2	ETS	3	5
MA0093	USF	bHLH-ZIP	4	6
MA0093	USF	bHLH-ZIP	1	3
MA0018	CREB	bZIP	5	7

Scope 3: 46 Values (Top 20 Shown)

PWM ID	PWM Name	Family	Position 1	Position 2
MA0076	SAP-1	ETS	3	6
MA0076	SAP-1	ETS	2	5
MA0076	SAP-1	ETS	4	7
MA0062	NRF-2	ETS	4	7
MA0028	Elk-1	ETS	6	9
MA0062	NRF-2	ETS	3	6
MA0062	NRF-2	ETS	1	4
MA0028	Elk-1	ETS	5	8
MA0018	CREB	bZIP	7	10
MA0076	SAP-1	ETS	5	8
MA0028	Elk-1	ETS	4	7
MA0058	Max	bHLH-ZIP	4	7
MA0062	NRF-2	ETS	5	8
MA0058	Max	bHLH-ZIP	5	8
MA0003	AP2alpha	AP2	1	4

MA0079	SP1	ZN-FINGER_C2H2	1	4
MA0048	Hen-1	bHLH	3	6
MA0058	Max	bHLH-ZIP	6	9
MA0079	SP1	ZN-FINGER_C2H2	4	7
MA0062	NRF-2	ETS	2	5

Scope 4: 34 Values (Top 20 Shown)

PWM ID	PWM Name	Family	Position 1	Position 2
MA0076	SAP-1	ETS	3	7
MA0076	SAP-1	ETS	2	6
MA0028	Elk-1	ETS	4	8
MA0088	Staf	ZN-FINGER_C2H2	8	12
MA0018	CREB	bZIP	4	8
MA0028	Elk-1	ETS	5	9
MA0058	Max	bHLH-ZIP	4	8
MA0088	Staf	ZN-FINGER_C2H2	7	11
MA0058	Max	bHLH-ZIP	5	9
MA0062	NRF-2	ETS	4	8
MA0093	USF	bHLH-ZIP	2	6
MA0014	Bsap	PAIRED	12	16
MA0062	NRF-2	ETS	3	7
MA0076	SAP-1	ETS	4	8
MA0018	CREB	bZIP	6	10
MA0042	HFH-3	FORKHEAD	7	11
MA0079	SP1	ZN-FINGER_C2H2	4	8
MA0052	MEF2	MADS	1	5
MA0062	NRF-2	ETS	6	10
MA0076	SAP-1	ETS	5	9

Scope 5: 12 Values

PWM ID	PWM Name	Family	Position 1	Position 2
MA0028	Elk-1	ETS	4	9
MA0042	HFH-3	FORKHEAD	7	12
MA0048	Hen-1	bHLH	2	7
MA0048	Hen-1	bHLH	6	11
MA0058	Max	bHLH-ZIP	4	9
MA0062	NRF-2	ETS	2	7
MA0076	SAP-1	ETS	2	7
MA0076	SAP-1	ETS	3	8
MA0079	SP1	ZN-FINGER_C2H2	3	8
MA0079	SP1	ZN-FINGER_C2H2	4	9
MA0088	Staf	ZN-FINGER_C2H2	7	12
MA0093	USF	bHLH-ZIP	1	6

Scope 6: 16 Values

PWM ID	PWM Name	Family	Position 1	Position 2
MA0028	Elk-1	ETS	4	10
MA0042	HFH-3	FORKHEAD	6	12
MA0048	Hen-1	bHLH	3	9
MA0050	Irf-1	TRP-CLUSTER	4	10

MA0051	Irf-2	TRP-CLUSTER	5	11
MA0052	MEF2	MADS	3	9
MA0058	Max	bHLH-ZIP	3	9
MA0062	NRF-2	ETS	2	8
MA0062	NRF-2	ETS	3	9
MA0062	NRF-2	ETS	4	10
MA0076	SAP-1	ETS	1	7
MA0076	SAP-1	ETS	2	8
MA0093	USF	bHLH-ZIP	1	7
MA0107	p65	REL	1	7
MA0107	p65	REL	3	9

Scope 7: 23 Values

PWM ID	PWM Name	Family	Position 1	Position 2
MA0014	Bsap	PAIRED	5	12
MA0018	CREB	bZIP	3	10
MA0027	EN-1	HOMEO	3	10
MA0031	FREAC-4	FORKHEAD	1	8
MA0033	FREAC-7	FORKHEAD	1	8
MA0038	Gfi	ZN-FINGER_C2H2	3	10
MA0042	HFH-3	FORKHEAD	4	11
MA0046	HNF-1	HOMEO	4	11
MA0047	HNF-3beta	FORKHEAD	4	11
MA0051	Irf-2	TRP-CLUSTER	9	16
MA0052	MEF2	MADS	1	8
MA0052	MEF2	MADS	2	9
MA0055	Myf	bHLH	2	9
MA0055	Myf	bHLH	5	12
MA0058	Max	bHLH-ZIP	3	10
MA0061	NF-kappaB	REL	2	9
MA0062	NRF-2	ETS	3	10
MA0065	PPARgamma-RXRal	NUCLEAR_RECEPTOR	9	16
MA0073	RREB-1	ZN-FINGER_C2H2	2	9
MA0088	Staf	ZN-FINGER_C2H2	8	15
MA0099	c-FOS	bZIP	1	8
MA0101	c-REL	REL	3	10
MA0107	p65	REL	2	9

Scope 8: 8 Values

PWM ID	PWM Name	Family	Position 1	Position 2
MA0052	MEF2	MADS	1	9
MA0052	MEF2	MADS	2	10
MA0068	Pax-4	PAIRED-HOMEO	3	11
MA0088	Staf	ZN-FINGER_C2H2	1	9
MA0088	Staf	ZN-FINGER_C2H2	7	15
MA0105	p50	REL	1	9
MA0106	p53	P53	6	14
MA0106	p53	P53	11	19

Transcription Factor Centric Discovery of Regulatory Elements in Mammalian Genomes Using Alignment-Independent Conservation Maps

Nilanjana Banerjee and Andrea Califano

Department of Biomedical Informatics, Columbia University
New York, New York 10032, USA
nb2188@genomecenter.columbia.edu, califano@c2b2.columbia.edu

Abstract. The computational identification of DNA binding sites that have high affinity for a specific transcription factor is an important problem that has only been partially addressed in prokaryotes and lower eukaryotes. Given the higher length of regulatory regions and the relative low complexity of DNA binding signature, however, methods to address this problem in higher order eukaryotes are lacking. In this paper, we propose a novel computational framework, which combines cellular network reverse engineering, integrative genomics, and comparative genomic approaches, to address this problem for a set of human transcription factors. Specifically, we study the regulatory regions of putative orthologous targets of a given transcription factor, obtained by reverse engineering methods, in several mammalian genomes. Highly conserved regions are identified by pattern discovery. Finally DNA binding sites are inferred from these regions using a standard Position Weight Matrices (PWM) discovery algorithm. By framing the identification of the PWM as an optimization problem over the two parameters of the method, we are able to discover known binding sites for several genes and to propose reasonable signatures for genes that have not been previously characterized.

Keywords: reverse engineering, comparative genomics, DNA binding site analysis, pattern discovery, transcriptional regulation, systems biology.

1 Introduction

Genome wide approaches have been increasingly successful in identifying regulatory interactions and in dissecting cellular networks from large microarray expression profile sets and other high throughput data modalities. However, the specific interactions between transcription factors (TFs) and their cognate cis regulatory elements ultimately responsible for network behavior, remain largely unmapped, especially in a mammalian context.

Transcription factor binding sites (TFBS) are often very degenerate and have low information content. In a mammalian context, where regulatory regions can

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 200–214, 2006.

be thousands of kilobases long, the probability of finding any given TFBS by chance, in a regulatory region is relatively high, making it difficult to assess enrichment or to identify sites that have a functional role in vivo. As a consequence, most recent advances in the field have resulted from increasingly sophisticated approaches based on integrative and comparative genomics methods. These either combine evidence from different clues (e.g. expression, ChIP-Chip data) or from different organisms where the regulatory modules of specific genes are likely to have been conserved through evolution [10].

Comparative genomics approaches, in particular, have been exceedingly helpful in increasing the specificity of regulatory elements identified by computational means. For instance, comparisons of human-mouse promoter regions of co-regulated muscle-specific genes, dramatically reduced the sequence search space, thus improving TFBS detection [24]. Several methods have further extended the use of comparative genomics methods in this context. For instance, [15] increased the specificity of Gibbs sampling methods by biasing the search in regions that are conserved in human-mouse and *C. elegans*-*C. briggsiae* genomes. Within a single organism, [25] successfully mapped several yeast binding sites by comparing aligned sequence profiles from several orthologous yeast genes. Finally, a few approaches have exploited knowledge of the phylogenetic distance among input species [3,23].

Moving from individual genes (or co-regulated gene sets) to entire genomes, a number of methods have also been proposed to identify over-represented motifs that are likely to play a functional role. Genome-wide multiple alignment and comparative analysis of several yeast species, for instance, yielded conserved and over-represented functional sequences, both coding and non-coding [8,26]. Similarly, comparisons of several vertebrate species provided systematic approaches to discover functional elements in the human genome [28,20].

A limitation of some comparative genomics methods is that they generally detect TFBSs in locally conserved regions, within pairwise or multiple sequence alignments of orthologous promoter sequences. This approach is less effective when these sequences are significantly divergent and may fail altogether if they have undergone genomic rearrangements that do not preserve the sequence order. To address some of these issues, the use of a non-alignment based method to discover conserved regulatory regions has been suggested [9]. However, this method is restricted to genome pairs.

Collectively, these methods have been increasingly successful in studying the conservation or enrichment of DNA binding sequences in the regulatory sequence of specific genes or even in genome wide fashion. However, by and large, they do not address the discovery of transcription factor specific binding sites an important open problem in biology. There are a few exceptions, which address this problem in *S.cerevisiae* [29]. These, however, are not easily extended to mammalian genomes for two reasons: (1) transcriptional regulation programs in mammalian cellular networks are significantly more complex and (2) mammalian regulatory regions are significantly longer, thus resulting in much higher background noise.

In this paper, we propose a novel framework which combines cellular network reverse engineering, integrative genomics, and comparative genomic approaches to discover the statistical profile of DNA binding sites that are highly specific for a given human TF, represented as Position Weight Matrices (PWM) [7]. Specifically, by using recent results on the reverse engineering of mammalian regulatory networks, based on the information-theoretic ARACNE algorithm [2,12,17], we identify high-probability putative targets of a human TF of interest. These have been shown to be enriched in biochemically validated targets of the TF. It is thus reasonable to expect that genes selected in this fashion will have regulatory regions that are also enriched for bona fide TFBS of the corresponding TF, thus increasing the signal to noise ratio. We show that for many TFs, such an enrichment is sufficient to correctly identify a TF-specific DNA binding signature by computational means.

Briefly, given a set of n_i putative target genes of a given transcription factor inferred by ARACNE, the algorithm proceeds through the following steps: first, for each target gene, the set of its orthologous regulatory regions from several mammalian genomes (Human, Chimpanzee, Mouse, Rat, and Dog) is assembled. This is called the Regulatory Region Set (RRS). Each RRS is then explored in isolations using the SPLASH motif discovery algorithm [5] to generate a Local Conservation Map (LCM) at a desired percent coverage, n_c. The LCM includes the most conserved $n_c\%$ of base pairs in the RRS. Then, the set of Local Conservation Maps for all the n_i putative targets, at a desired percent coverage n_c, is assembled into a Global Conservation Map (GCM). Finally, the GCM is analyzed using the DME motif discovery algorithm [22] to identify statistically significant Position Weight Matrices (PWM). The PWMs are optimized over all possible values of (n_t, n_c), and the most statistically significant PWMs are reported. This process is schematically illustrated in Fig. 1. Finally, reported PWMs are compared to known TF motif in TRANSFAC, if available, or validated via the literature. Pattern discovery based conservation maps are introduced to overcome the limitations of alignment methods which may fail in regions where overall conservation is low or where translocation events may have produced a non alignable, yet overall conserved sequence.

We first test our approach on a set of six TFs with known TFBS, including *MYC*, *E2F1*, *TFDP1*, *IRF7*, *FOSL1* and *NFkB2*. Targets were obtained by running ARACNE on a set of 254 microarray expression profiles of normal and tumor related human B cells. To assess the decrease in algorithm performance when ARACNE targets are used, as opposed to biochemically validated ones, we also run the procedure on a set of validated *MYC* targets. Results show that our approach is able to accurately identify the TFBS of 4 of the 6 genes (*MYC*, *E2F1*, *TFDP1* and *IRF7*) at a relatively low percent coverage. Furthermore, we find that while the *MYC* PWM, which identifies the E Box CACGTG, is more statistically significant when identified from the biochemically validated targets, the E Box PWM is still clearly the most significant even when only ARACNE targets are used.

We then apply our method to two TFs whose TFBS have not yet been fully characterized. These include *BCL6* and *HOXD13*. The putative PWM for *BCL6* matches a binding site identified by in vitro *BCL6* binding assays . We also report that the canonical PWM for *FOSL1* and *NFkB2* could not be recovered as the most statistically significant ones. This is likely because these genes are modulated at the post-translational level and thus ARACNE fails to provide a set of putative targets that is significantly enriched in true targets of the TF. For each TF we provide the set of genes that gives the most significant PWM and the associated percent coverage.

This paper introduces a new comparative genomics approach that uses an alignment-independent motif-discovery strategy to identify conserved elements given a set of putative TF targets. Unlike most current approaches, it is not fixed to an arbitrary percent coverage defined by global alignment. We find that, in most cases, the method successfully discovers the PWM for a specific human TF and the related high-confidence functional targets.

2 Methods

Our procedure is aimed at identifying high-affinity DNA binding sites for a given human TF. We do so, by first selecting TF targets using the ARACNE reverse engineering method, and then by integrating the information contained in the mammalian orthologous promoters of the candidate targets into a global conservation map (GCM). The GCM is then analyzed using a conventional position weight matrix discovery algorithm [22]. The steps, which are schematically shown in Figure 1, include:

1. Selection of set of n_t putative targets of a given TF
2. Identification of orthologous regulatory regions in mammalian genomes
3. Generation of target-specific, local conservation maps (LCM) at a predefined percent coverage, n_c
4. Generation of a global conservation map (GCM) from the set of all LCMs,
5. Identification of the most statistically significant PWMs by motif discovery in the GCM, and
6. optimization of the PWM enrichment p value across all values of the parameters (n_t, n_c).

Details of each step are given below.

2.1 Selection of Putative Targets of a Transcription Factor

Our method assumes that a candidate pool of targets of a specific human TF is available. In the context of this paper these were generated either by mining the literature data (only for the set of biochemically validated targets of *MYC*) or by network reverse engineering using the ARACNE algorithm [2,17]. Specifically, the bootstrap version of ARACNE was used [18], which improves the quality of the predictions with respect to the original version of the algorithm. Targets

were inferred from a phenotypically rich collection of 254 microarray expression profiles of normal and tumor related human B lymphocytes, also described in [18]. Briefly, the bootstrap version of ARACNE builds a consensus network by running the algorithm on a large number ($n = 100$) of distinct sets of 254 expression profiles. These are obtained by sub sampling the original dataset with replacement. Interactions detected in at least k independent runs of the algorithm are reported, with k chosen so that that the probability of a false positive interaction (p value) is less than 5%, after multiple hypothesis testing. The p-value is based on a null-hypothesis obtained by randomly permuting the inferred edges in each bootstrapped network.

The genes corresponding to the 50 most statistically significant interactions with each TF of choice are selected and ranked by the consensus score. Targets that are positively and negatively correlated with the TF are assembled in separate sets and analyzed separately by our method, since the specific DNA binding mechanism associated with target activation or repression by a TF is usually distinct. For instance, target activation by MYC is mediated by an E-Box while target repression by MYC, while not completely understood, is not mediated by an E Box.

2.2 The Regulatory Region Sets

Given a rank-ordered set of n_t targets of a human TF (either validated or putative), each target gene g_i^{TF} is studied in isolation. Its coding sequence is first aligned in the human, chimpanzee, dog, mouse and rat genomes. Subsequently the transcription start site (TSS) is identified from the genome annotation in each such sequence. When the TSS is not annotated for one organism, the corresponding human mRNA, up to the TSS, is aligned with the organisms genome to determine a putative location. The [2kb, +2kb] non-coding and repeat-masked region (relative to the TSS) in each mammalian genome is finally selected as an orthologous promoter. The set of such sequences for a specific target gene is called a Regulatory Region Set, $RRS_i^{TF} = \{g_{ij}^{TF}, j = hs, ..., rr\}$, where the index j indicates the specific genome. As additional well annotated mammalian genomes become available, they can be easily incorporated into this step of the analysis, thus further improving the signal to noise ratio.

2.3 Generating Gene-Specific, Local Conservation Maps

Each RSS_i^{TF} set is then analyzed using the SPLASH motif discovery algorithm to discover a set of statistically significant sparse motifs that are conserved across at least three species, including *H. sapiens*. Reported motifs must have a minimum density of eight matching nucleotides in any window of ten nucleotides and at least 8 matching nucleotides. Since SPLASH does not set any limits on the maximum motifs length, very long (and thus highly statistically significant) sparse motifs may emerge in regions with relatively high cross species conservation. The SPLASH discovery parameters were extensively studied and

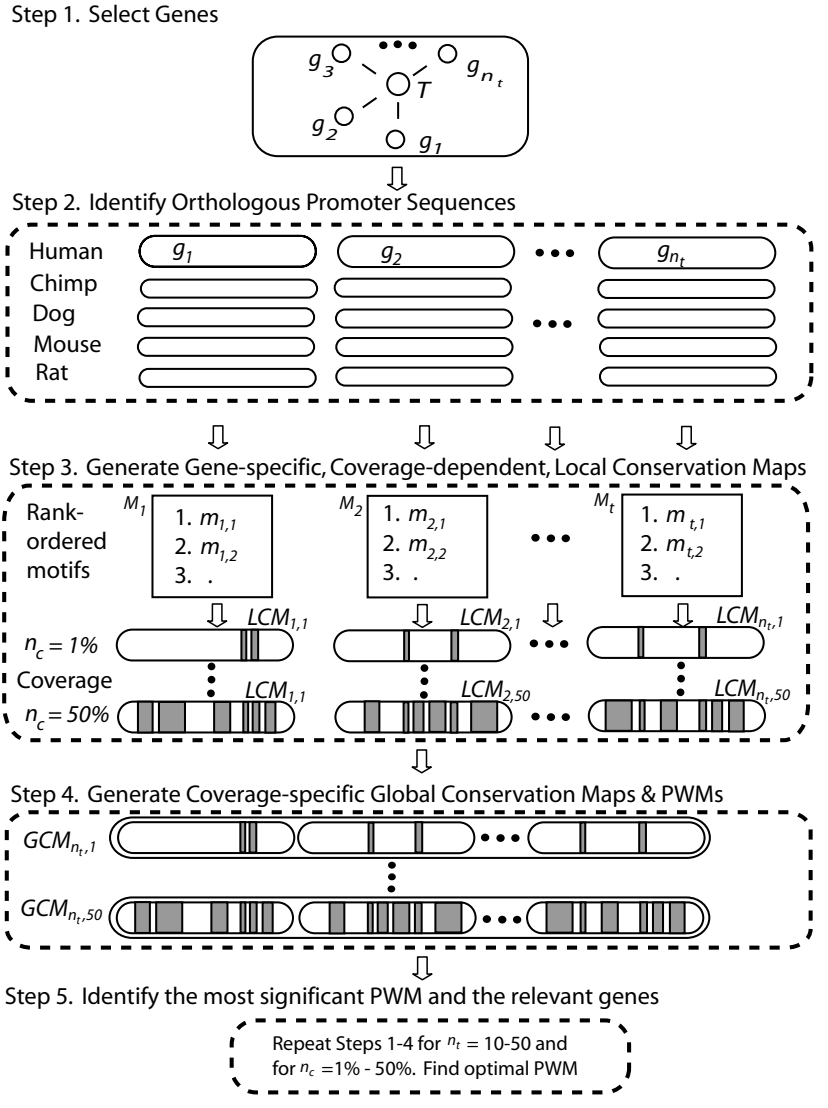


Fig. 1. Schematic diagram of our TF-centric approach to discover regulatory elements using alignment-independent conservation maps. Step 1 identifies candidate targets of a TF by ARACNE bootstrapping. In Step 2, the orthologous promoter sequences are extracted. Step 3 generates gene-specific conservation maps varying over percent coverage. Step 4 compiles a global conservation map over all targets with varying percent coverage. The process is repeated for varying number of top-ranked targets. Finally, Step 5, identifies the most significant PWM for a TF the relevant target genes and the associated percent coverage.

optimized for the *MYC* transcription factor. Sparser motif density (e.g. 6 conserved nucleotides in any window of 10) resulted in an excessively large number of reported motifs, thus increasing background noise. Higher densities (e.g. 8 conserved nucleotides in any window of 9 or greater) resulted in very few motifs, thus increasing the likelihood that regions containing important regulatory elements may be missed. While a complete study of the SPLASH parameter space for each TF is outside the boundary of this paper, this can be accomplished by using sets of validated biological targets for a few known TFs. For this paper, we optimized these parameters using the biochemically validated targets of *MYC* and then we use these optimal parameters for the discovery of the PWM of the other TFs. In particular, the same SPLASH parameters were used both to analyze the biochemically validated targets of *MYC* and the ARACNE predicted targets of all the studied TFs.

The SPLASH motifs M_i identified from a given RRS are then ranked based on their statistical significance. Statistical significance is essentially a function of the motif length (number of conserved nucleotides), nucleotide composition (based on the frequency of each nucleotide in the set of sequences), and number of orthologous sequences containing the motif.

Given the k most statistically significant SPLASH motifs for a Regulatory Region Set, RSS_i^{TF}, a human-centric Local Conservation Map LCM_i^{TF} can be trivially generated by selecting only the nucleotides covered by these motifs in the RRS. We define the percent coverage n_c of the Local Conservation Map as the ratio between the total number of nucleotides in the map and the total number of nucleotides in the full Regulatory Region Set. For instance, a 50% coverage ratio indicates that approximately one half of the nucleotides of the original Regulatory Region Set are retained in the Local Conservation Map. Clearly, a variable percent coverage can be achieved by selecting a different number k of significant motifs. The maximum possible coverage is achieved when additional motifs are no longer statistically significant at the desired p value level or when all the SPLASH motifs are exhausted. Conversely, given a desired percent coverage (as long as it is not too large), a number of motifs k can be chosen such that the closest coverage value is achieved in the Local Conservation Map. Thus, for each desired coverage, n_c, two local conservation maps can be created from each transcription factor TF, $LCM_i^{TF+}(n_c)$ and $LCM_i^{TF-}(n_c)$, corresponding respectively to the activated and to the repressed (putative) targets of the TF.

2.4 Generating a Global Conservation Map

Given a TF, a percent coverage n_c, and a number of putative targets n_t, the local conservation maps for all the putative human targets activated or repressed by the TF ($LCM_1^{TF\pm}(n_c)$,...,$LCM_{nt}^{TF\pm}(n_c)$) are pooled together into two Global Conservation Maps for the transcription factor, $GCM^{TF+}(n_c, n_t)$ and $GCM^{TF-}(n_c, n_t)$. A variety of different methods can then be used to identify binding sites that are enriched in either GCM, compared to a background of random promoters. We used the Discriminative Matrix Enumerator method (DME) [22] to search for over-represented PWMs spanning respectively six, eight

and nine nucleotides. DME is especially suited to detect matrices that may be highly degenerate and have sparse occurrences. We ranked the list of PWMs generated by DME based on statistical significance relevant to our context. The statistical significance of each discovered PWM is evaluated based on the expected vs. actual number of matches of the PWM in the Global Conservation Map, at the specific coverage n_c. The expected background frequency of the motif (null hypothesis) is determined from a set of 9,000 non-coding promoters representing all the unique genes on the HU95Av2 Affymetrix chip used to generate the expression data. The statistical significance is then determined using the hypergeometric distribution.

2.5 Identification of the Most Significant PWM and the Relevant Target Genes

If the putative targets detected by ARACNE are enriched in true targets and if a PWM is functionally significant, a trend should emerge where the statistical significance of the PWM first increases as a function of n_t (as more true targets are added) and then starts to decrease as more and more false positives start to be included. Similarly, as we change the percent coverage, n_c, we should see a similar trend since for low n_c values bona fide sites may be missed, while for large values of n_c the background noise becomes dominant.

For each Global Conservation Map, $GCM^{TF\pm}$ (n_c, n_t), we thus determine the values of $n_t = n_t^{*1}, n_t^{*2}$ and $n_t = n_t^{*1}, n_t^{*2}$ yielding the two most statistically significant, non-identical PWMs and report the corresponding PWMs, PWM_{*1}^{TF} and PWM_{*2}^{TF}. These are then compared against known PWM for the TF or further validated as putative profiles for a novel binding site.

2.6 Comparison of PWMs

To compare two matrices, the KL-divergence score is computed in all possible relative reference frames. The score is zero for identical matrices; the more different the two matrices, the higher the KL-divergence [21]. Values below 1.0 reflect substantial identity of the matrix composition.

3 Results

In this section, we report the optimal PWMs identified by our approach for eight TFs. We chose six TFs whose binding sites have been well-characterized (*MYC*, *E2F1*, *TFDP1*, *IRF7*, *FOSL1* and *NFkB2*) and two TFs which do not have well-characterized binding sites (*BCL6* and *HOXD13*). A key requirement is that these TFs must have a sufficient number of ARACNE inferred targets to support our statistical analysis. We arbitrarily set that threshold at 50 targets but we find that the correct matrix could be identified with as few as 10 targets. We studied the positively-correlated and negatively-correlated target genes separately as they might lead to clues to distinct mechanisms of transcription

Table 1. Statistically significant PWMs discovered for TFs with a previously known PWM and TFs with a previously unknown PWM. Analysis was performed both on TF+ *positively − correlated* and TF-*negatively − correlated* targets. TF- analysis was not performed when there were not sufficient targets i.e. TFDP1-, FOSL1-, NFKB2-.

TF	Size	Motif 1	Pvalue	Match TF	Match Matrix ID	Score	Motif 2	Pvalue	Match TF Match TF	Match Matrix ID	Score
MYC+	w6	CACGTG	1.00E-07	MYC	M00118	0	TCCGGM	0.00008.4	ELK1	M00025	0.726
	w8	SCACGTGS	1.00E-08	MYC	M00118	0.2	GCCATGGB	1.00E-06	RFX1	M00281	0.72
MYC-	w6	CAGCKG	1.00E-08				CGCWGC	2.29E-07			
	w9	DSYGCWGCG	4.40E-10				CGTCCWBGT	2.50E-08			
E2F1+	w6	AGGGSC	1.00E-08				ACGTGM	1.00E-05	MYC-MAX	M00018	0.969
	w8	VTCACGTG	7.90E-10	MYC	M00123	0.65	GCGCVAAA	1.20E-08	E2F1	M00428	0.55
	w9	ARGGGMSCV	5.60E-12				KWCWKGGCS	2.20E-11			
E2F1-	w6										
	w9	GAAKCKGVY	2.00E-10				CDRTCCGHG	5.00E-10			
TFDP1+	w6	TTSGCG	2.20E-05	TFDP/E2F 1	M00431	0.35	GTTTTM	1.00E-04	MEIS1 HOXA9	M00420	0.89
	w8	GCGCBAAA	1.00E-07	TFDP/E2F1	M00516	0.23					
	w9	BTTVGCGCS	0.00E+00	TFDP/E2F1	M00516	0.259					
IRF7+	w6	AAACYG	1.00E-06	IRF7	M00453	0.975					
	w8	GAAANTGA	8.80E-08	IRF7	M00453	0.79					
	w9	TTCNDTTTC	1.00E-10	IRF7	M00453	0.55	ARBCTRCTG	1.00E-06	IRF7	M00453	0.55
IRF7-	w6	CCGSCA	9.90E-06	LXR_DR4	M00766	0.72	na	18 and 4			
	w9	RGCGSCGMY	2.00E-10				CCGCKCSDS	1.00E-09			
FOSL1+	w6	GGWCCC	1.00E-07	LRF	M01100	0.89	CTKCCC	1.00E-06			
	w9	SGGCDGGRS	1.00E-09	SP1	M00196	0.83	CCTSCSKBC	1.00E-10			
NFKB2+	w6	GGSCCC	1.00E-07								
	w8	CCYKMCCC		SP1	M00196	0.7					
	w9	CMCCWKGGB					SCCHWGGKG				
BCL6+	w6										
	w9	AKAAAAWVA	1.00E-09				CCSSSCCCM	1.00E-08	SP1	M00933	0.9
BCL6-	w6	NA					NA				
	w9	AKCGNCGMT	5.00E-09				TTYCYAGRM	1.00E-09			
HOXD13+	w6	CWGCCC	1.00E-10				GGGBCC	7.00E-10			
	w9	CYCCHGSAS					GRHKCTGGS				
HOXD13-	w6	ACCTGS	1.50E-07				CATCGW	6.00E-06			
	w9	GDSVAGGTG	1.00E-09	AREB6	M00413	0.843	CACCTNSYC	1.00E-08	AREB6	M00413	0.83

regulation. For each regulatory region set (RSS) we report the two most significant PWMs and show that in most cases the correct PWM is reported as one of these two matrices (often the most significant one).

We compare the discovered PWMs to TRANSFAC (version 10.1) and JASPAR databases (version 2004) and found that for four of the six tested TFs (*MYC*, *E2F1*, *TFDP1*, *IRF7*) the discovered PWM matched the known ones at high specificity ($s < 1.0$) in the expected set of targets (activated or repressed). For instance, for MYC the correct PWM was found in the activated targets, which is known to be mediated by an E-Box.

Interestingly, most of the other discovered PWMs (i.e. those not matching the TF) match the profile of some other well-characterized binding sites in TRANSFAC or JASPAR, revealing possible combinatorial regulation mechanisms. For two transcription factors (*NFkB2* and *FOSL1*) none of the discovered PWMs matched the known one. We present and discuss some of the DNA binding profiles in more detail. The complete set of discovered PWMs, matched matrices and significance values are shown in Table 1.

3.1 Transcription Factors with a Previously Known PWM

MYC: We tested our approach on the positively correlated (activated) and negatively correlated (repressed) putative targets of the *MYC* proto oncogene. Analysis of the positively-correlated targets yields the well-known binding site (E-box) as the optimal PWM for both tested motif lengths (w = 6 and w = 8). The optimal statistical significance was obtained for $n_t = 30$ top-ranked putative targets by ARACNE and n_c =12). Figure 2 shows the statistical significance of the PWM enrichment as a function of both n_t and n_c. The second most significant motif of length 6 matches the known *ELK1* motif. *ELK1* is a member of the ETS oncogene family. Recently, researchers have suggested that a novel ETS transcription factor TEL2 and MYC cooperate in development of many human B-cell malignancies [4]. This analysis suggests that *ELK1* may play a similar combinatorial role in the transcriptional regulation of some *MYC* targets expressed in B cells. The second most significant motif of word length 8 matched to *RFX1* which has been implicated in transcriptional downregulation of *MYC* [19] and may thus form a feed-forward loop with *MYC*.

As expected, the two most significant PWMs (both for lengths 6 and 8) from the negatively correlated *MYC* targets do not match the E-box nor any other known DNA binding motif. This suggests that repression by MYC is mediated by a distinct and yet to be defined transcriptional regulation mechanism. The discovered matrices may thus help elucidate such a mechanism.

We also performed this analysis with a set of validated *MYC* targets (see Supplemental Information). As expected, the significance value is high even for $n_t = 10$ target genes and increases modestly as more validated genes are included (Figure 3).

E2F1: Analysis of the positively correlated E2F1 targets shows that the most significant PWMs match the known binding profiles of *MYC* and E2F1. The PWMs were discovered with target genes and n_c% coverage. These two

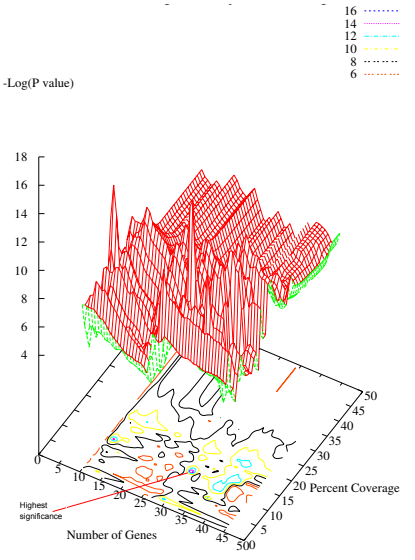


Fig. 2. Significance of a discovered PWM as a function of number of genes and percent coverage in ARACNE predicted targets of MYC ($n_t = 30$ and $n_c = 12$)

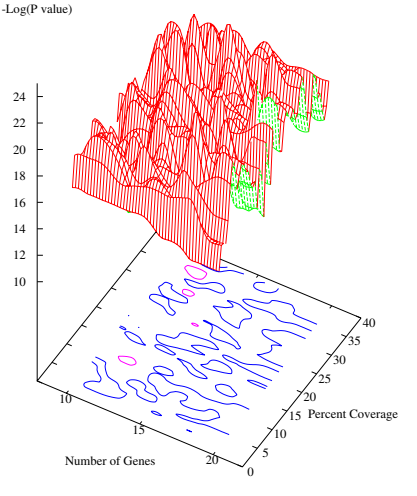


Fig. 3. Significance of a discovered PWM as a function of number of genes and percent coverage in validated targets of MYC. There is a steady increase in significance as more validated targets are added to the gene set ($n_c = 20$ and $n_t = 21$).

transcription factors have been known to cooperate in activation of human cancers [11]. E2F1 is known to act as a repressor in some cases. As expected, the analysis of negatively-correlated targets does not yield PWMs that match

any previously characterized motifs. The method suggests two alternate motifs GAAKCKGVY ($n_t = 25$ and $n_c = 16$) and CDRTCCGHG ($n_t = 26$ and $n_c = 32$) as possible binding sites for this mechanism.

TFDP1: E2F factors bind to DNA as homodimers or heterodimers in association with dimerization partner DP1. TFDP1 is an example of such a family of related transcription factors. Not surprisingly, the BTTVGCGCS motif discovered by the analysis from the positively-correlated target genes, with $n_t = 30$ and $n_c = 5$, is a close match to the previously known motifs for both *E2F1* and TFDP1. Very few targets were predicted to be negatively-correlated targets of TFDP1 and were thus not suited for our analysis.

IRF7: We identified the known binding motif for IRF7 as the top motif for word lengths 6, 8 and 9. For word length 9, the PWM was identified with $n_t = 11$ and $n_c = 30\%$. Analysis on the negatively-correlated targets of IRF7 yielded PWMs with matches to several ubiquitous TFs e.g. SP1 and LXR-alpha.

FOSL1 and *NFkB2*: We were unable to identify the known motifs of both FOSL1 and NFKB2. Most likely, the ARACNE inferred targets of these two genes are not sufficiently enriched in their true transcriptional targets. This is generally the case when the transcriptional regulation of a gene is significantly mediated by post-translational events and thus the mRNA concentration does not correlate with target expression.

3.2 Transcription Factors Without a Previously Unknown PWM

BCL6: BCL6 is a transcriptional repressor whose activity is important in the creation of the Germinal Center and in formation of non-Hodgkins B cell lymphomas [6]. Thus, as expected, the PWMs discovered in the positively-correlated targets do not match any known TRANSFAC or JASPAR motif. Analysis of the negatively-correlated targets, on the other hand, reveals that the second most significant PWM of length w = 9 (TTYCYAGRM) , with $n_t = 20$ and $n_c = 29$ closely matches a known BCL6 binding site discovered from in vitro binding assays [6]. We have identified a subset of the ARACNE-predicted BCL6 targets based on the strength of correlation and number of hits of the discovered motif. Chromatin immunoprecipitation assays are being performed to test whether these genes are true targets of BCL6.

HOXD13: HOXD13 belongs to the homeobox family of TFs. In mammals, knowledge of the genetic pathways, including the possible direct or indirect targets, regulated by HOX proteins is extremely limited [27]. Analysis on the positively-correlated targets of HOXD13 did not yield a close match to previously characterized motifs. However, the negatively correlated targets allowed the identification of a PWM (GDSVAGGTG with $n_t = 28$ and $n_c = 17$), which is very similar to that of AREB6, a zinc-finger homeodomain enhancer-binding protein. Further analysis of this PWM can possibly help understand the details of HOXD13 protein function.

4 Discussion

Discovery of DNA binding profiles that are specific to a given TF is still an open problem in mammalian biology. Specifically, we are unaware of any previous method that has been successful in the identification of the binding site of a specific TF in a mammalian context.

We report a systems biology approach that combines network reverse engineering, comparative genomics, and bioinformatics analysis which is successful in identifying the correct binding site for a significant number of the tested TFs. The method is computationally intensive, requiring several hours to fully explore the parameter space and to identify the optimal PWM. As a result, a genome wide analysis of all transcription factors exceeds the scope of this paper. Thus, we limit our analysis to 8 representative transcription factors. Six of these (*MYC*, *E2F1*, *TFDP1*, *IRF7*, *FOSL1* and *NFkB2*) have a known DNA binding profile in either TRANSFAC or JASPAR, while the remaining 2 (*BCL6* and *HOXD13*) are not well characterized although a few instances of the DNA binding site are known for *BCL6*.

We show that for 4 of the 6 previously characterized transcription factors, our approach is successful in discovering PWMs that closely match the previously known profiles. For *BCL6* we show that the inferred PWM matches a previously known binding site instance. In addition, for *HOXD13* negatively-correlated targets we identify a PWM that matches (albeit with some variability) with the known motif for another homeodomain protein.

The new approach innovates on previous methods in several ways: while sets of co-regulated genes were used before to identify binding sites, this is the first method that relies on a network reverse engineering algorithm (ARACNE) to identify putative targets of a known transcription factor for further analysis. This allows to causally link the discovered profile with a specific transcription factor. Additionally, rather than relying on sequence alignment, the method uses pattern discovery to identify regions that are highly conserved across orthologous promoter regions in several mammalian genomes. Finally, rather than using ad-hoc parameters, the method systematically explores the parameter space to discover the most statistically significant PWMs. This optimization step is critical, as it is difficult to find a single set of parameters that would perform uniformly across the set of tested transcription factors.

The method has several limitations and could be improved in several ways. First of all, if the set of putative targets is not enriched in bona fide targets of the TF, then DNA binding motif discovery is compromised. This is especially likely to happen for transcription factors that are constitutively expressed and regulated at the post-translational level. For these, the mRNA concentration of the TF is not strongly correlated to the activation/repression of its targets and thus the ARACNE method would likely produce putative target set that are not sufficiently enriched in bona fide targets. Thus, any improvements in the reverse engineering of transcriptional networks will directly translate into improved performance of the proposed method. A second issue is that the algorithm may identify binding sites associated with co-factors that may be equally

or even more enriched in the specific subset of targets generated by ARACNE. As a result, we recommend that the top two or three PWMs, for each tested length (w = 6, 8) are considered and validated as candidate binding site profiles. Conversely, the method may be useful in identifying co factor binding sites, suggesting a possible cooperative regulation mechanism.

Acknowledgments. We thank Andrew Smith, Kai Wang, Wei Keat Lim and Hans-Erik Aronson for expert assistance.

References

1. Altschul, S., Erickson, B.: Significance of nucleotide sequence alignments: A method for random sequence permutation that preserves dinucleotide and codon usage. Mol. Biol. Evol. **2** (1985) 528-538
2. Basso, K., Margolin, A. A., Stolovitzky, G., Klein, U., Dalla-Favera, R., Califano, A.: Reverse engineering of regulatory networks in human B cells. Nat. Genetics **37** (2005) 382-390
3. Blanchette, M., Tompa, M.: Discovery of regulatory elements by a computational method for mphylogenetic footprinting. Genome Research **12** (2002) 739-748
4. Cardone, M., Kandilci, A.: The Novel ETS Factor TEL2 Cooperates with Myc in B Lyemphomagenesis. Molecular and Cellular Biology **25** (2005) 2395-2405
5. Califano, A.: SPLASH: structural pattern localization analysis by sequential histograms. Bioinformatics. Bioinformatics **16** (2000) 341-357
6. Chang, C., Ye, B., Chaganti, R., Dalla-Favera, R.: BCL6, a POZ/zinc-finger protein, is a sequence-specific transcriptional repressor. PNAS **93** (1996) 6947-6952
7. Claverie, J.: Some useful statistical properties of position-weight matrices. Comput. Chemistry **18** (1994) 287-294
8. Cliften, P., Sudarsanam, P., Desikan, A., Fulton, L., Fulton, B., Majors, J., Waterston, R., Cohen, B.A., Johnston, M.: Finding functional features in Saccharomyces genomes by mphylogenetic footprinting. Science **301** (2003) 71-76
9. Elemento, O., Tavazoie S.: Fast and systematic genome-wide discovery of conserved regulatory elements using a non-alignment based approach. Genome Biol. **6** (2005) R18
10. Harbison, C.T., Gordon, D.B., Lee, T.I., Rinaldi, N.J., Macisaac, K.D., Danford, T.W., Hannett, N.M., Tagne, J.B., Reynolds, D.B., Yoo, J., Jennings, E.G., Zeitlinger, J., Pokholok, D.K., Kellis, M., Rolfe, P.A., Takusagawa, K.T., Lander, E.S., Gifford, D.K., Fraenkel, E., Young, R.A.: Transcriptional regulatory code of a eukaryotic genome. Nature. **431** (2004) 99-104
11. Hong, S., Pusapati, R.V., Powers, J.T., Johnson, D.G.: Oncogenes and the DNA Damage Response: Myc and *E2F1* Engage the ATM Signaling Pathway to Activate p53 and Induce Apoptosis. Cell cycle **5** (2005) 801-803
12. Hartemink, A.J.: Reverse engineering gene regulatory networks. Nature Biotechnology **23** (2005) 554-555
13. Kharchenko, P., Vitkup, D., Church, G.M.: Filling gaps in a metabolic network using expression information. Bioinformatics **20** (2000) I178-I185
14. Lenhard, B., Sandelin, A., Mendoza, L., Engstrm, P., Jareborg, N., Wasserman, W.: Identification of conserved regulatory elements by comparative genome analysis J. Biol **2** (2003) 13

15. Liu, Y., Liu, X.S., Wei, L., Altman, R.B., Batzoglou, S.: Eukaryotic regulatory element conservation analysis and identification using comparative genomics. Genome Res. **3** (2004) 451-458
16. Docquier, F,, Farrar, D., D'Arcy, V., Chernukhin, I., Robinson, A.F., Loukinov, D., Vatolin, S., Pack, S., Mackay, A., Harris, R.A., Dorricott, H., O'Hare, M.J., Lobanenkov, V., Klenova, E.: Heightened expression of CTCF in breast cancer cells is associated with resistance to apoptosis Cancer Research **65** (2005) 5122-5125
17. Margolin, A.A., Nemenman I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R., Califano, A.: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context. BMC Bioinformatics **7** (2006) S7.
18. Margolin, A.A., Wang, K., Lim, W.K., Kustagi, M., Nemenman, I., Califano A.: Reverse engineering cellular networks. Nature Protocols **1** (2006) 662-671
19. Ohashi, Y., Ueda, M., Kawase, T., Kawakami, Y. Toda, M.: Identification of an epigenetically silenced gene, RFX1, in human glioma cells using restriction landmark genomic scanning. Oncogene **23** (2004) 7772-7779
20. Prakash, A., Tompa, M.: Discovery of regulatory elements in vertebrates through comparative genomics. Nature Biotechnology **102** (2005) 14689-14693
21. Schones, D., Sumazin, P., Zhang, M.Q.: Similarity of position frequency matrices for transcription factor binding sites. Bioinformatics **21** (2005) 307-313
22. Smith, A., Sumazin, P., Zhang, M.Q.: Identifying tissue-selective transcription factor binding sites in vertebrate promoters. PNAS **102** (2005) 1560-1565
23. Sinha, S., Blanchette, M., Tompa, M.: PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. BMC Bioinformatics **5** (2004) 170
24. Wasserman, W.W., Palumbo, M., Thompson, W. Fickett, J.: Human-mouse genome comparisons to locate regulatory sites. Nature Genetics **26** (2000) 225-228.
25. Wang, T., Stormo, G.: Combining phylogenetic data with co-regulated genes to identify regulatory motifs. Bioinformatics **18** (2003) 2369-2380.
26. Wang, T., Stormo, G.: Identifying the conserved network of cis-regulatory sites of a eukaryotic genome. PNAS **102** (2006) 17400-17405.
27. Williams, T., Williams, M., Kuick, R., Misek, D., McDonagh, K., Hanash, S., Innis, J.: Candidate downstream regulated genes of HOX group 13 transcription factors with and without monomeric DNA binding capability. Developmental Biology **279** (2005) 462-480
28. Xie, X., Lu, J., Kulbokas, E.J., Golub, T.R., Mootha, V., Lindblad-Toh, K., Lander, E.S., Kellis, M.: Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals. Nature **434** (2005) 338-345
29. Zhu, Z., Pilpel, Y., Church, G.M.: Computational identification of transcription factor binding sites via a transcription-factor-centric clustering (TFCC) algorithm. J. Mol. Biol. **318** (2002) 71-81

Identifiability Issues in Phylogeny-Based Detection of Horizontal Gene Transfer

Cuong Than[1], Derek Ruths[1], Hideki Innan[2], and Luay Nakhleh[1,⋆]

[1] Dept. of Computer Science, Rice University, Houston, TX, USA
[2] Human Genetics Center, The University of Texas Health Science Center, Houston, TX, USA
nakhleh@cs.rice.edu

Abstract. Prokaryotic organisms share genetic material across species boundaries by means of a process known as *horizontal gene transfer* (HGT). Detecting this process bears great significance on understanding prokaryotic genome diversification and unraveling their complexities. Phylogeny-based detection of HGT is one of the most commonly used approaches for this task, and is based on the fundamental fact that HGT may cause gene trees to disagree with one another, as well as with the species phylogeny. Hence, methods that adopt this approach compare gene and species trees, and infer a set of HGT events to reconcile the differences among these trees.

In this paper, we address some of the identifiability issues that face phylogeny-based detection of HGT. In particular, we show the effect of inaccuracies in the reconstructed (species and gene) trees on inferring the correct number of HGT events. Further, we show that a large number of maximally parsimonious HGT scenarios may exist. These results indicate that accurate detection of HGT requires accurate reconstruction of individual trees, and necessitates the search for more than a single scenario to explain gene tree disagreements. Finally, we show that disagreements among trees may be a result of not only HGT, but also *lineage sorting*, and make initial progress on incorporating HGT into the coalescent model, so as to stochastically distinguish between the two and make an accurate reconciliation. This contribution is very significant, particularly when analyzing closely related organisms.

1 Introduction

Whereas eukaryotes evolve mainly though lineal descent and mutations, bacteria obtain a large proportion of their genetic diversity through the acquisition of sequences from distantly related organisms via horizontal gene transfer (HGT); e.g., see [4,19]. There has been a big "ideological and rhetorical" gap between the researchers believing that HGT is so rampant that a prokaryotic phylogenetic tree is useless and those who believe HGT is mere "background noise" that does not affect the reconstructibility of a phylogenetic tree for bacterial genomes.

⋆ Corresponding author.

G. Bourque and N. El-Mabrouk (Eds.): RECOMB-CG 2006, LNBI 4205, pp. 215–229, 2006.

Supporting arguments for these two views have been published. For example, the heterogeneity of genome composition between closely related strains (only 40% of the genes in common with three *E. coli* strains [29]) supports the former view, whereas the well-supported phylogeny reconstructed by Lerat *et al.* from about 100 "core" genes in γ-Proteobacteria [13] gives evidence in favor of the latter view. Nonetheless, regardless of the views and the accuracy of the various analyses, there is a consensus as to the occurrence of HGT and the evolutionary role it plays in bacterial genome diversification. Further, HGT is a main process by which bacteria develop resistance to antibiotics (e.g., [5]), is considered a primary explanation of incongruence among gene phylogenies, and is a significant obstacle to reconstructing the Tree of Life [3].

The HGT detection problem concerns the detection of the genes that are horizontally transferred into the genome, the donors and recipients of every horizontally transferred gene, and the number of HGT events that occurred during the evolutionary history of a set of species. When HGT occurs, the evolutionary history of the gene(s) involved does not necessarily agree with that of the species phylogeny. This observation is the fundamental basis of the phylogeny-based HGT detection approach: trees for individual genes are reconstructed (and sometimes a species tree is reconstructed as well, using other data), and their disagreements are identified to estimate the number (how many) as well as locations (donors and recipients) of HGT events. Beside the computationally challenging problem of quantifying disagreements among trees for the sake of detecting HGT, major challenges that face this approach include (1) determining whether the disagreements are indeed due to HGT, and (2) whether there is a unique HGT "scenario". Yet, these two challenges encompass a host of issues of which we address three. First, since trees are at best partially known, they have to be reconstructed using a phylogeny reconstruction method. We investigate the impact that the quality of reconstructed trees has on HGT detection. Second, under the assumption that HGT is actually the source of tree disagreements, we investigate the uniqueness of a solution to the HGT detection problem. Finally, among closely related species, *lineage sorting* due to random genetic drift may also cause tree incongruence, thus mimicking the effects of HGT on phylogenies. In this case, accurate HGT detection requires determining the actual cause of tree incongruities, and making the appropriate reconciliation. We make preliminary progress on incorporating HGT into the coalescent model, so as to produce a stochastic framework for classifying population-level events (such as lineage sorting) and species-level events (such as HGT).

We draw several conclusions from this work. First, to obtain accurate estimates of HGT based on tree incongruence, poorly supported edges of reconstructed trees should be removed; this is a hard task, but is very important to conduct. Second, eliminating statistical error from reconstructed trees leads to non-binary trees, and hence phylogeny-based HGT detection methods should be designed to handle such trees (rather than focus on binary trees, which many existing tools do). Third, more than one maximally parsimonious solution (a solution that has the minimum number of HGT edges, or events, to

explain the species and gene tree incongruence) may exist, and hence HGT detection methods should search for all such solutions. Finally, trees may be incongruent due to processes other than HGT; hence, classifying the sources of incongruence and reconciling them accordingly is imperative.

2 Tree Incongruence and HGT Detection

A gene tree is a model of how a gene evolves. As a gene at a locus in the genome replicates and its copies are passed on to more than one offspring, branching points are generated in the gene tree. Because the gene has a single ancestral copy, barring recombination, the resulting history is a branching tree [14]. Thus, within a species, many tangled gene trees can be found, one for each nonrecombined locus in the genome. Exploring incongruence among gene trees is the basis for phylogeny-based HGT detection and reconstruction.

We illustrate some of the scenarios that may lead to gene tree incongruence in Figure 1. The species tree is represented by the "tubes"; it has A and B as sister taxa whose most recent common ancestor (MRCA) is a sister taxon of C.

In the case of HGT, shown in Figure 1(a), genetic material is transferred from one lineage to another. Sites that are not involved in a horizontal transfer are inherited from the parent while other sites are horizontally transferred from another species. Figure 1(b) gives an example of a gene tree that disagrees with the species phylogeny because of lineage sorting due to random genetic drift: the genes of B and C coalesced before their MRCA coalesced with the gene of species A. Moreover, sometimes multiple events "cancel out" one another's effects when co-occurring in the same dataset; for example, in Figure 1(c), lineage sorting "hides" the incongruence between the species and gene trees (tree topologies) that would have resulted from the HGT event. Another factor that may lead to gene and species tree disagreements is that trees reconstructed by phylogenetic methods may not be completely accurate (we refer to this as *statistical error* in the trees); hence, disagreements among trees due to such inaccuracies may trigger HGT "signal", thus leading to overestimation of the actual HGT events.

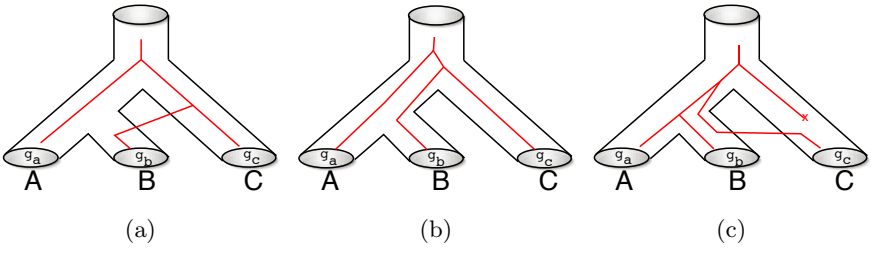


Fig. 1. (a) Gene tree that disagrees with the species tree due to (a) HGT from C to B and (b) lineage sorting due to random genetic drift. In (c), the effect of the HGT event (from B to C) is "canceled out" by random genetic drift, resulting in congruent species and gene trees.

Notice that in the case of lineage sorting, the species phylogeny is still a tree, and the gene trees should be reconciled within its branches. However, in the case of HGT, the evolutionary history of the species genomes may not be represented by phylogenetic trees; rather, *phylogenetic networks* are the appropriate model [16,12]. The phylogeny-based HGT detection problem seeks the phylogenetic network with minimum number of *reticulation nodes*, e.g., HGT edges, to reconcile the species and gene trees. The minimization simply reflects a maximally parsimonious solution: in the absence of any additional biological knowledge, the simplest solution is sought. In the case, the simplest solution is one that invokes the minimum number of HGT events to explain tree incongruence. There has been a large body of work on this problem; e.g., see [7,18,2,17,15].

3 The Effect of Statistical Error on HGT Detection

In this section we investigate, through simulations, the effect of error in the reconstructed trees on the detection of HGT. In particular we consider the minimum number of HGT events inferred by HGT detection methods, as well as the number of such maximally parsimonious solutions found by these methods.

Experimental Setting. We used the `r8s` tool [25] to generate four random birth-death phylogenetic trees, T_i, $i \in \{10, 25, 50, 100\}$, where i denotes the number of taxa in the tree. The `r8s` tool generates molecular clock trees; we deviated the trees from this hypothesis by multiplying each edge in the tree by a number randomly drawn from an exponential distribution. The expected evolutionary diameter (longest path between any two leaves in the tree) is 0.2. Then, from each model "species" tree T_i, we generated five different "gene" trees, $T_{i,j}$, $j \in \{1, 2, 3, 4, 5\}$, where j denotes the number of *subtree prune and regraft* (SPR) moves applied to T_i to obtain $T_{i,j}$.[1] For each T_i and $T_{i,j}$, $i \in \{10, 25, 50, 100\}$ and $j \in \{1, 2, 3, 4, 5\}$, and for each sequence length $\ell \in \{250, 500, 1000, 2000, 4000, 8000\}$, we generated 30 DNA sequence alignments $S_i^\ell[k]$ and $S_{i,j}^\ell[k]$, $1 \leq k \leq 30$, whose evolution was simulated down their corresponding trees under the GTR+Γ+I (gamma distributed rates, with invariable sites) model of evolution, using the Seq-gen tool [20]. We used the parameter settings of [30]. Then, from each sequence alignment, we reconstructed a tree TNJ using the Neighbor Joining (NJ) method [24], and another tree using a maximum parsimony heuristic as implemented in PAUP* [26]. Since the maximum parsimony heuristic may return a set of optimal trees, for each alignment we only considered the *strict consensus* of each such set, and referred to that as the tree TMP. At the end of this process we had 4 trees T_i, 20 trees $T_{i,j}$, 720 NJ trees $TNJ_i^\ell[k]$, 3600 NJ trees $TNJ_{i,j}^\ell[k]$, 720 MP trees $TMP_i^\ell[k]$, and 3600 MP trees $TMP_{i,j}^\ell[k]$ ($i \in \{10, 25, 50, 100\}$, $j \in \{1, 2, 3, 4, 5\}$, $1 \leq k \leq 30$, and $\ell \in \{250, 500, 1000, 2000, 4000, 8000\}$). To compute minimal HGT scenarios as well as the number of such scenarios, we applied two methods to pairs of species and gene trees: LatTrans [7,1] and RIATA-HGT [17] (we modified the latter

[1] An SPR move simulates an HGT event.

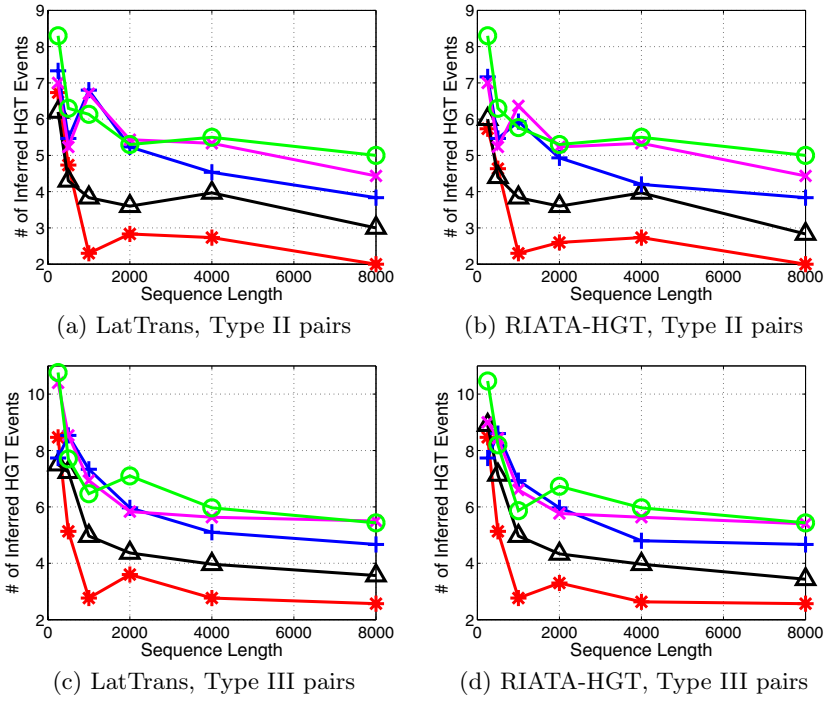
tool so that it computes multiple solutions, rather than a single solution as was originally described by the authors). Both tools were applied to three different types of pairs of trees.

Type I pairs $(T_i, T_{i,j})$**:** in this case, the species and gene trees are assumed to be correct.
Type II pairs $(T_i, TNJ^{\ell}_{i,j}[k])$ and $(T_i, TMP^{\ell}_{i,j}[k])$**:** in this case, the species tree is correct, and the gene trees are estimated (using NJ and MP, respectively).
Type III pairs $(TNJ^{\ell}_{i}[k], TNJ^{\ell}_{i,j}[k])$ and $(TMP^{\ell}_{i}[k], TMP^{\ell}_{i,j}[k])$**:** in this case, both the species and gene trees are inferred.

The goal of running the methods in these different ways is to estimate the error due to inaccuracy in the different trees. Due to space limitations, we only show results using NJ trees, 25-taxon trees (Since LatTrans cannot handle non-binary trees, it was not run on MP trees). In each run of a tool on a pair of trees, we computed two values: the number of inferred HGT events, and the number of such scenarios (or solutions) found by the method. In Type II and Type III pairs, we report the average of all 30 runs for each combination of i, j, and ℓ.

3.1 The Effect of Statistical Error on Estimating the Number of HGT Events

Both LatTrans and RIATA-HGT computed the correct number of SPR moves (i.e., HGT edges) when applied to Type I pairs. In other words, when both the species and gene trees were correct, both methods made an accurate estimation of the number of HGT events. The performance of both methods, in terms of the number of inferred HGT events, on Type II and Type III pairs of trees is shown in Fig. 2. Figs. 2(a) and 2(b) show that when the species tree is accurate, and the gene tree is inferred, both methods accurately estimate the number of HGT events for the case of 5 HGT events when the sequences are of length 8000. They overestimate the number for all other cases, at all sequence lengths. As the sequence length increases, the trees inferred by NJ become more accurate, since NJ is *statistically consistent*, and hence the improvement in the performance of the methods as the sequence length increases. At sequence length 250, the methods have the worst performance. When both the species and gene trees are inferred, the overestimation becomes larger, as shown in Figs. 2(c) and 2(d). In this case, even at sequence length 8000 the methods do overestimate the actual number of HGT events. It is worth noting that both methods have almost identical performance in terms of the number of HGT events inferred (RIATA-HGT does slightly better in some cases at sequence length 1000). However, RIATA-HGT is orders of magnitude faster. Given that the two methods accurately estimated the number of HGT events in Type I pairs of trees, i.e., accurate species and gene trees, the results show that error in inferred trees (one or both) leads to overestimation of the number of HGT events. The overestimation is even larger for the larger data sets (50- and 100-taxon trees). Therefore, it is important to



(a) LatTrans, Type II pairs (b) RIATA-HGT, Type II pairs

(c) LatTrans, Type III pairs (d) RIATA-HGT, Type III pairs

Fig. 2. The number of HGT events inferred by LatTrans and RIATA-HGT, as a function of the sequence length. Each curve corresponds to one of the five actual numbers of HGT events: $\star$: 1 HGT; $\triangle$: 2 HGTs; $+$: 3 HGTs; $\times$: 4 HGTs; and $\circ$: 5 HGTs. 25-taxon trees inferred using NJ.

eliminate statistical error from trees before estimating HGT events. Ruths and Nakhleh [23] have studied the performance of various methods for eliminating wrong edges while maintaining accurate ones. This elimination, in the form of contracting poorly supported edges, leads to non-binary trees, which cannot be handled by LatTrans, although they can be handled by RIATA-HGT. Therefore, a second conclusion is that phylogeny-based HGT detection methods should be designed to handle both bi- and multi-furcating trees.

3.2 The Uniqueness of HGT Scenarios

Moret *et al.* [16] showed that a phylogenetic network that reconciles two trees need not be unique, by showing two phylogenetic networks with a single reticulation event that reconcile the same pair of trees. Further, they showed how branch lengths could be used to resolve the non-uniqueness question in this simple case. Here we show that the number of possible maximally parsimonious (with minimum number of HGT events) phylogenetic networks that reconcile a pair of trees may actually be exponential. Further, we discuss when branch lengths may not be sufficient to resolve the non-uniqueness issue.

The number of maximally parsimonious HGT scenarios that reconcile a pair of trees (species and gene trees, for example) may be exponentially large, as illustrated in Fig. 3. The species and gene trees in the figure, ST and GT, respectively, contain $3k$ leaves and differ in that X_{i2} is closer to X_{i1} than to X_{i3} in tree ST, and closer to X_{i3} than to X_{i1} in tree GT, for $1 \leq i \leq k$. For every triplet $\langle X_{i1}, X_{i2}, X_{i3} \rangle$ of taxa, one of three HGT edges is needed to reconcile the difference in topologies of the triplet based on the two trees ST and GT: (1) the edge $H_{i1} : X_{i3} \rightarrow X_{i2}$, (2) the edge $H_{i2} : X_{i2} \rightarrow X_{i3}$, or (3) the edge $H_{i3} : m_i \rightarrow X_{i1}$, where m_i is the edge incoming into the most recent common ancestor (node) of the triplet of taxa; these three scenarios are shown in Fig. 4. To reconcile the differences among all k triplets, there are 3^k HGT scenarios, since there are k triplets to reconcile, and for each triplet there are three possible reconciliations. Two observations are in order. First, since the

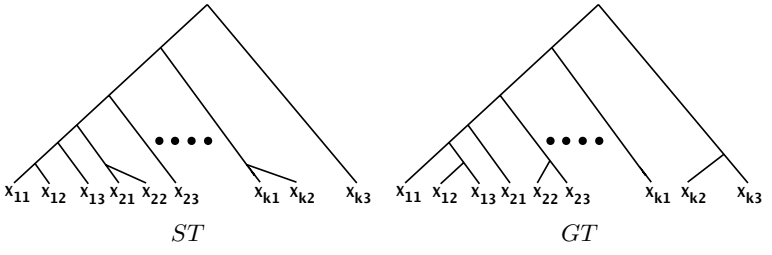


Fig. 3. A species tree ST and a gene tree GT with $3k$ leaves. The two trees differ in k places: the species tree has X_{i1} and X_{i2} as siblings, whereas the gene tree has X_{i2} and X_{i3} as siblings ($1 \leq i \leq k$). There are 3^k maximally parsimonious HGT scenarios that reconcile the two trees.

donor and recipient of a gene have to co-exist in time [16], and given that the topology of a phylogeny defines a partial order on the set of extant and ancestral taxa (ancestral taxa *precede* their descendants in this partial order), it follows that edge H_{i3} can be part of an HGT solution only if certain taxa went extinct or were not sampled. This case is illustrated in Fig. 4, where the dashed line represents the lineage for taxon X_i which is not present in the set of taxa under consideration but whose existence must be invoked to explain the HGT edge H_{i3}.

Let δ_{ST} and δ_{GT} be the pairwise distance matrices of the set of taxa based on the species and gene trees ST and GT, respectively, in Fig. 3, and let us consider the triplet of taxa in Fig. 4. There are three cases. (1) The scenario H_{i1} is plausible if and only if $\delta_{ST}(X_{i1}, X_{i3}) \approx \delta_{GT}(X_{i1}, X_{i3})$ and $\delta_{ST}(X_{i1}, X_{i2}) \not\approx \delta_{GT}(X_{i1}, X_{i2})$. (2) The scenario H_{i2} is plausible if and only If $\delta_{ST}(X_{i1}, X_{i2}) \approx \delta_{GT}(X_{i1}, X_{i2})$. (3) The scenario H_{i3} is plausible if and only if If $\delta_{ST}(X_{i2}, X_{i3}) \approx \delta_{GT}(X_{i2}, X_{i3})$. Since the conditions in the three cases are mutually exclusive, it follows the branch lengths, when estimated accurately, can be used to correctly resolve the non-uniqueness issue in this case. However, estimating branch lengths

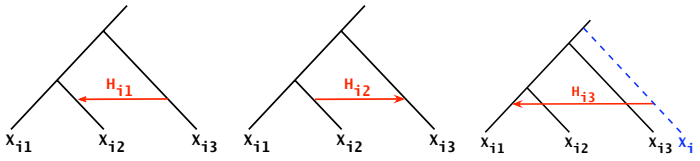


Fig. 4. The three possible scenarios for reconciling the topologies of the triplet $\langle X_{i1}, X_{i2}, X_{i3} \rangle$ based on the species and gene trees, ST and GT respectively, in Fig. 3.

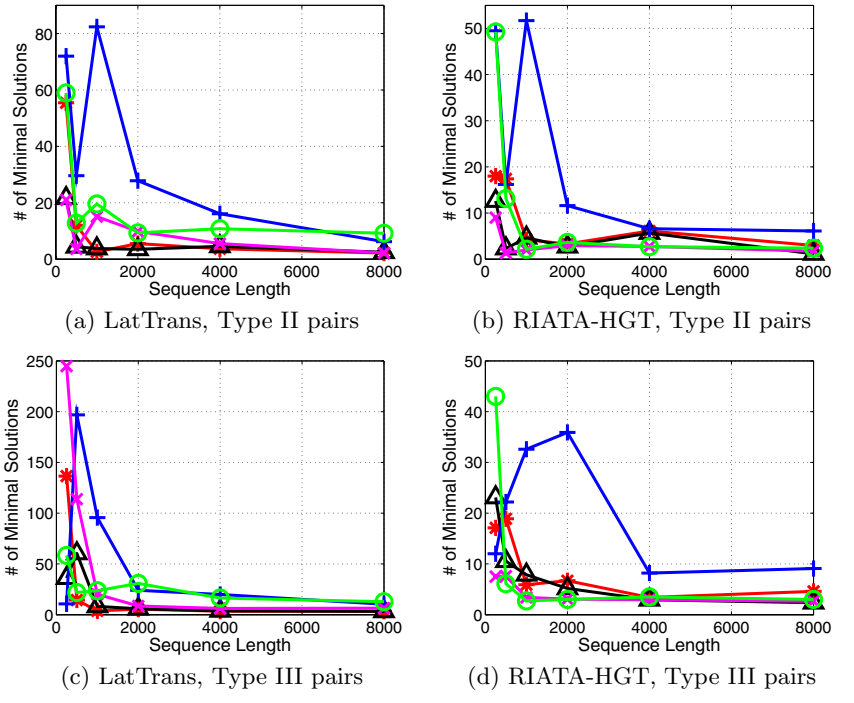
to a high degree of accuracy such that the above three cases are distinguished accurately is a very challenging task. Further, even if branch lengths are estimated accurately, if the evolutionary distance between the donor and recipient is very small, distinguishing among the cases becomes more challenging.

In our simulation study, we looked at the number of maximally parsimonious solutions that were computed by LatTrans and RIATA-HGT; the results for 25-taxon NJ trees are shown in Fig. 5. All fours graphs show that, regardless of whether the actual or inferred species trees are used, both methods estimate a large number of maximally parsimonious solutions. The figures show that the number decreases as the sequences used become longer. When we ran the methods on the actual trees (Type I pairs of trees), both of them returned single solutions. A plausible conclusion is that as the amount of statistical error in the inferred trees increases, so does the number of maximally parsimonious solutions. The reason for behind this is that for shorter sequence lengths, the accuracy of the trees is poorer, i.e., they have more wrong edges. These wrong edges give an indication of more HGT events. This indication, though false, leads to larger numbers of solutions since more reconciliations become possible. The peaks around sequence lengths 1000 and 2000 in Fig. 5 coincide with the peaks in Fig. 2, which gives an indication that as the number of inferred HGT events increases, so does the number of possible solutions. An important conclusion is that phylogeny-based HGT detection methods should be designed to compute "all" possible solutions. As illustrated in Fig. 3, the number of such solutions may be exponential, though. A measure that assigns support to these solutions is imperative, so that they can be rank ordered.

4 Incorporating HGT into the Coalescent

As we described in Section 2, phylogenetic incongruence may occur due to various processes, of which HGT is only one. Another such process is lineage sorting, whose effect and confusing signal to HGT detection is particularly important when analyzing genes of closely related organisms. In this section, we augment the coalescent model by incorporating HGT, thus providing a framework for stochastically distinguishing among these two processes as the actual source of phylogenetic incongruence.

Lineage sorting occurs because of random contribution of each individual to the next generation. Some fail to have offsprings while some happen to have



(a) LatTrans, Type II pairs (b) RIATA-HGT, Type II pairs

(c) LatTrans, Type III pairs (d) RIATA-HGT, Type III pairs

Fig. 5. The number of minimal HGT scenarios inferred by LatTrans and RIATA-HGT as a function of the sequence length. Each curve corresponds to one of the five actual numbers of HGT events: $\star$: 1 HGT; $\triangle$: 2 HGTs; $+$: 3 HGTs; $\times$: 4 HGTs; and $\circ$: 5 HGTs. 25-taxon trees inferred using NJ.

multiple offsprings. In population genetics, this process was first modeled by R. A. Fisher and S. Wright, in which each gene of the population at a particular generation is chosen independently from the gene pool of the previous generation, regardless of whether the genes are in the same individual or in different individuals. Under the Wright-Fisher model, "the coalescent" considers the process backward in time [11,9,27]. That is, the ancestral lineages of genes of interest are traced from offsprings to parents. A coalescent event occurs when two (or sometimes more) genes are originated from the same parent, which is called the most recent common ancestor (MRCA) of the two genes.

The basic process can be treated as follows. Consider a pair of genes at time τ_1 in a random mating haploid population. The population size at time τ is denoted by $N(\tau)$. The probability that the pair are from the same parental gene at the previous generation (time $\tau_1 + 1$) is $1/N(\tau_1 + 1)$. Therefore, starting at τ_1, the probability that the coalescence between the pair occurs at τ_2 is given by

$$Prob(\tau_2) = \frac{1}{N(\tau_2)} \sum_{\tau=\tau_1+1}^{\tau_2-1} \left(1 - \frac{1}{N(\tau)}\right). \quad (1)$$

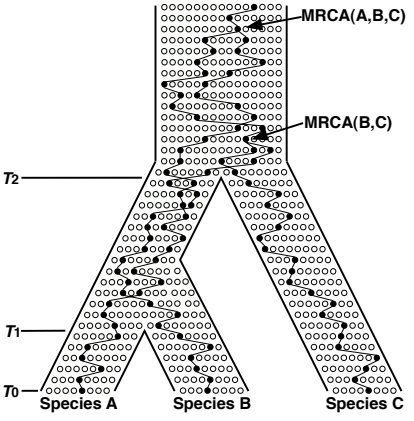
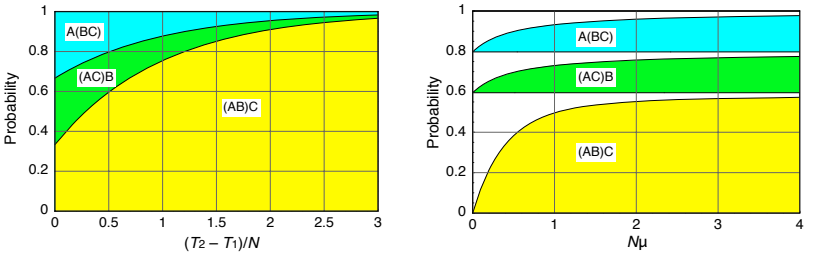


Fig. 6. An illustration of the coalescent process in a three species model with discrete generations. The process is considered backward in time from present, T_0, to past. Circles represent haploid individuals. We are interested in the gene tree of the three genes (haploids) from the three species. Their ancestral lineages are represented by closed circles connected by lines. A coalescent event occurs when a pair of lineages happen to share a single parental gene (haploid).

When $N(\tau)$ is constant, the probability density distribution (pdf) of the coalescent time (i.e., $t = \tau_2 - \tau_1$) is given by a geometric distribution, and can be approximated by an exponential distribution for a large N:

$$Prob(t) = \frac{1}{N} e^{-t/N}. \tag{2}$$

The coalescent process is usually ignored in phylogenetic analysis, but has a significant effect (causing lineage sorting) when closely related species are considered [8,28,21]. The situation of Fig. 1(b) is reconsidered under the framework of the coalescent in Fig. 6. Here, it is assumed that species A and B split $T_1 = 5$ generations ago, and the ancestral species of A and B and species C split $T_2 = 19$ generation ago. The ancestral lineage of a gene from species A and that from B meet in their ancestral population at time $\tau = 6$, and they coalesce at $\tau = 35$, which predates T_2, the speciation time between (A, B) and C. The ancestral lineage of B enters in the ancestral population of the three species at time $\tau = 20$, and first coalesces with the lineage of C. Therefore, the gene tree is represented by $A(BC)$ while the species tree is $(AB)C$. That is, the gene tree and species tree are "incongruent". Under the model in Fig. 6, the probability that the gene tree is congruent with the species tree is 0.85, which is one minus the product of the probability that the ancestral lineages of A and B do not coalesce between $\tau = 6$ and $\tau = 9$, and the probability that the first coalescence in the ancestral population of the three species occur between (A and C) or (B and C). The former probability is $\frac{14}{15}\frac{12}{13}\frac{11}{12}...\frac{7}{8}\frac{7}{8} = 0.22$ and the latter is $\frac{2}{3}$.



(a) (b)

Fig. 7. (a) The probabilities of the three types of gene tree, (AB)C, (AC)B, and A(BC), as functions of $(T_2 - T_1)/N$. (b) The probabilities that the gene tree is resolved from DNA sequence data. The probabilities are given as functions of the mutation rate for the three types of tree, (AB)C, (AC)B, and A(BC), when $(T_2 - T_1)/N = 0.5$. The white regions represent the probabilities that the gene tree is not resolved.

Under the three-species model (Fig. 6), there are three possible types of gene tree, $(AB)C$, $(AC)B$ and $A(BC)$. Let $Prob$[(AB)C], $Prob$[(AC)B] and $Prob$[A(BC)] be the probabilities of the three types of gene tree. These three probabilities are simply expressed with a continuous time approximation when all populations have equal and constant population sizes, N, where N is large:

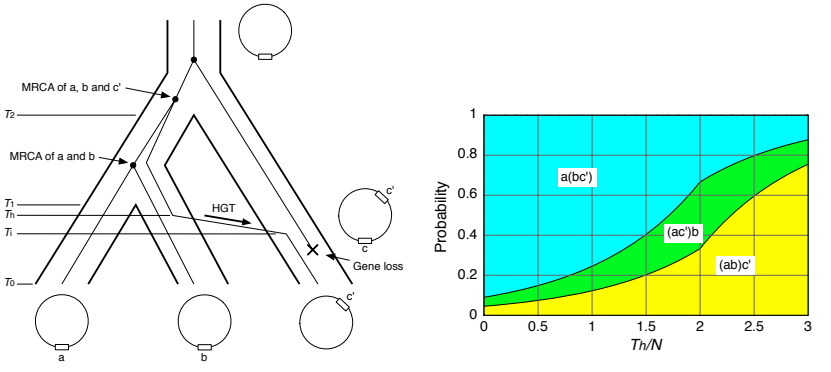
$$Prob[(\mathrm{AB})\mathrm{C}] = 1 - \frac{2}{3}e^{-(T_2-T_1)/N}, \tag{3}$$

and

$$Prob[(\mathrm{AC})\mathrm{B}] = Prob[\mathrm{A}(\mathrm{BC})] = \frac{1}{3}e^{-(T_2-T_1)/N}. \tag{4}$$

Figure 7(a) shows the three probabilities as functions of $(T_2 - T_1)/N$.

It is important to notice that the estimation of the gene tree from DNA sequence data is based on the nucleotide differences between sequences, and that the gene tree is sometimes unresolved. One of the reasons for that is a lack of nucleotide differences such that DNA sequence data are not informative enough to resolve the gene tree. This possibility strongly depends on the mutation rate. Let μ be the mutation rate per region per generation, and consider the effect of mutation on the estimation of the gene tree. We consider the simplest model of mutations on DNA sequences, the infinite site model [10], in which mutation rate per site is so small that no multiple mutations at a single site are allowed. Consider a gene tree, $(AB)C$, and suppose that we have a reasonable outgroup sequence such that we know the sequence of the MRCA of the three sequences. It is obvious that mutations on the internal branch between the MRCA of the three and the MRCA of A and B are informative. If at least one mutation occurred on this branch, the gene tree can be resolved from the DNA sequence alignment. This effect is investigated by assuming that the number of mutations on a branch with length t follows a Poisson distribution with mean μt. Fig. 7(b) shows the probability that the gene tree is resolved; $T_2 - T_1 = 0.5N$ generations is assumed



(a) (b)

Fig. 8. (a) A three bacterial species model with an HGT event. A demonstration that a congruent tree could be observed even with HGT. (b) The probabilities of the three types of gene tree, (ab)c', (ac')b, and a(bc'), as functions of T_h/N. $T_1 = 2N$ and $T_2 = 3N$ are assumed.

so that the probability that the gene tree is $(AB)C$ is about 0.6. As expected, as the mutation rate increases, the probability that the gene tree is resolved from the sequence alignment increases, and this probability exceeds 90% when $N\mu > 1.52$. Similar results are obtained for the other two types of trees, $(AC)B$ and $A(BC)$, that appears with probability 0.2 for each (see also Fig. 7(b)).

Thus far, we have shown that the gene tree is not always identical to the species tree even considering verical evolution. With keeping this in mind, let us consider the effect of horizontal gene transfer (HGT) on gene tree under the framework of the coalescent. The application of the coalescent theory to bacteria is straightforward. Rather than the Wright-Fisher model, bacterial evolution may be better described by the Moran model, which handles overlapping generations well. Suppose that each haploid individual in a bacterial population with size N has a lifespan that follows an exponential distribution with mean l. When an individual dies, another individual randomly chosen from the population replaces it to keep the population size constant. In other words, one of the $N - 1$ alive lineages is duplicated to replace the dead one. Under the Moran model, the ancestral lineages of individuals of interest can be traced backward in time, and the coalescent time between a pair of individuals follows an exponential distribution with mean $lN/2$ [6,22]. This means that one half of the mean lifetime in the Moran model corresponds to one generation in the Wright-Fisher model. It may usually be thought that HGT can be detected when the gene tree and species tree are incongruent (see Section 2). However, the situation is complicated when lineage sorting is also involved. Consider a model with three species, A, B, and C, in which an HGT event occurs from species B to C. Suppose the ancient circular genome has a single copy of a gene as illustrated in Fig. 8(a). Let a, b and c be the focal orthologous genes in the three species, respectively. At time T_h, a gene escaped from species B and was inserted in a genome in species

C at T_i, which is denoted by c'. Following the HGT event, c was physically deleted from the genome, so that each of the three species currently has a single copy of the focal gene. If there is no lineage sorting, the gene tree should be $a(bc')$. Since this tree is incongruent with the species tree, $(AB)C$, we could consider it as an evidence for HGT. However, as shown in Section 2, lineage sorting could also produce the incongruence between the gene tree and species tree without HGT. It is also important to note that lineage sorting, coupled with HGT, could produce congruent gene tree, as illustrated in Fig. 8(a). Although b and c' have a higher chance to coalesce first, the probability that the first coalescence occurs between a and b or between a and c' may not be negligible especially when $T_1 - T_h$ is short. The probabilities of the three types of gene tree can be formulated under this tri-species model with HGT as illustrated in Fig. 8(a). Here, T_h could exceed T_1, in such a case it can be considered that HGT occurred before the speciation between A and B. Assuming that all populations have equal (constant) population sizes, N, the three probabilities can be obtained modifying (3) and (4):

$$Prob[(\text{AB})\text{C}] = \begin{cases} \frac{1}{3}e^{-(T_1-T_h)/N} & \text{if } T_h \leq T_1 \\ 1 - \frac{2}{3}e^{-(T_h-T_1)/N} & \text{if } T_h > T_1 \end{cases}, \tag{5}$$

$$Prob[(\text{AC})\text{B}] = \begin{cases} \frac{1}{3}e^{-(T_1-T_h)/N} & \text{if } T_h \leq T_1 \\ \frac{1}{3}e^{-(T_h-T_1)/N} & \text{if } T_h > T_1 \end{cases}, \tag{6}$$

and

$$Prob[\text{A}(\text{BC})] = \begin{cases} 1 - \frac{2}{3}e^{-(T_1-T_h)/N} & \text{if } T_h \leq T_1 \\ \frac{1}{3}e^{-(T_h-T_1)/N} & \text{if } T_h > T_1 \end{cases}. \tag{7}$$

Fig. 8(b) shows the three probabilities assuming $T_1 = 2N$ and $T_2 = 3N$.

5 Conclusions and Future Work

In this paper, we showed that error in inferred trees has a negative impact on the estimates made by phylogeny-based HGT detection methods. These results provide a set of conclusions. First, to obtain accurate estimates of HGT based on tree incongruence, poorly supported edges of reconstructed trees should be removed; this is a hard task, but is very important to conduct. Second, eliminating statistical error from reconstructed trees leads to non-binary trees, and hence phylogeny-based HGT detection methods should be designed to handle such trees (rather than focus on binary trees, which many existing tools do). Third, more than one maximally parsimonious solution (a solution that has the minimum number of HGT edges, or events, to explain the species and gene tree incongruence) may exist, and hence HGT detection methods should search for all such solutions. In this preliminary work, we have studied the effect of error in inferred trees on the accuracy of HGT detection methods, both in terms of the minimum number of events computed as well as the number of such minimal solutions. One of our immediate goals is to study the performance of these

methods in terms of the locations (donors and recipients) of inferred HGT; for this task, we will use the distance measures proposed in [16].

Further, lineage sorting due to the coalescent process works as a noise for detecting and reconstructing HGT based on tree incongruence, sometimes mimicking the evidence for HGT and sometimes creating a false negative "evidence" for HGT. Therefore, to distinguish HGT and lineage sorting, a stochastic framework based on the theory introduced in Section 4 is needed. We only considered very simple cases with three species here, and we will extend the theory to more general cases.

References

1. L. Addario-Berry, M.T. Hallett, and J. Lagergren. Towards identifying lateral gene transfer events. In *Proc. 8th Pacific Symp. on Biocomputing (PSB03)*, pages 279–290, 2003.
2. M. Bordewich and C. Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics*, pages 1–15, 2005. In press.
3. V. Daubin, N.A. Moran, and H. Ochman. Phylogenetics and the cohesion of bacterial genomes. *Science*, 301:829–832, 2003.
4. W.F. Doolittle, Y. Boucher, C.L. Nesbo, C.J. Douady, J.O. Andersson, and A.J. Roger. How big is the iceberg of which organellar genes in nuclear genomes are but the tip? *Phil. Trans. R. Soc. Lond. B. Biol. Sci.*, 358:39–57, 2003.
5. I.T. Paulsen *et al.* Role of mobile DNA in the evolution of Vacomycin-resistant Enterococcus faecalis. *Science*, 299(5615):2071–2074, 2003.
6. W.J. Ewens. *Mathematical Population Genetics.* Springer-Verlag, Berlin, 1979.
7. M.T. Hallett and J. Lagergren. Efficient algorithms for lateral gene transfer problems. In *Proc. 5th Ann. Int'l Conf. Comput. Mol. Biol. (RECOMB01)*, pages 149–156, New York, 2001. ACM Press.
8. R. R. Hudson. Testing the constant-rate neutral allele model with protein sequence data. *Evolution*, 37:203–217, 1983.
9. R.R. Hudson. Properties of the neutral allele model with intergenic recombination. *Theor. Popul. Biol.*, 23:183–201, 1983.
10. M. Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61:893–903, 1969.
11. J. F. C. Kingman. The coalescent. *Stochast. Proc. Appl.*, 13:235–248, 1982.
12. V. Kunin, L. Goldovsky, N. Darzentas, and C.A. Ouzounis. The net of life: reconstructing the microbial phylogenetic network. *Genome Research*, 15:954–959, 2005.
13. E. Lerat, V. Daubin, and N.A. Moran. From gene trees to organismal phylogeny in prokaryotes: The case of the γ-proteobacteria. *PLoS Biology*, 1(1):1–9, 2003.
14. W.P. Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997.
15. V. Makarenkov. T-REX: Reconstructing and visualizing phylogenetic trees and reticulation networks. *Bioinformatics*, 17(7):664–668, 2001.
16. B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):13–23, 2004.

17. L. Nakhleh, D. Ruths, and L.S. Wang. RIATA-HGT: A fast and accurate heuristic for reconstrucing horizontal gene transfer. In L. Wang, editor, *Proceedings of the Eleventh International Computing and Combinatorics Conference (COCOON 05)*, pages 84–93, 2005. LNCS #3595.
18. L. Nakhleh, T. Warnow, and C.R. Linder. Reconstructing reticulate evolution in species–theory and practice. In *Proc. 8th Ann. Int'l Conf. Comput. Mol. Biol. (RECOMB04)*, pages 337–346, 2004.
19. H. Ochman, J.G. Lawrence, and E.A. Groisman. Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405(6784):299–304, 2000.
20. A. Rambaut and N. C. Grassly. Seq-gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comp. Appl. Biosci.*, 13:235–238, 1997.
21. N. Rosenberg. The probability of topological concordance of gene trees and species tree. *Theoretical Population Biology*, 61:225–247, 2002.
22. N.A. Rosenberg. Gene genealogies. In C.W. Fox and J. B. Wolf, editors, *Evolutionary Genetics: Concepts and Case Studies*, chapter 15. Oxford Univ. Press University Press, 2005.
23. D. Ruths and L. Nakhleh. Techniques for assessing phylogenetic branch support: A performance study. In *Proceedings of the Fourth Asia-Pacific Bioinformatics Conference (APBC 06)*, pages 187–196, 2006.
24. N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.
25. M. Sanderson. `r8s` software package. Available from http://loco.ucdavis.edu/r8s/r8s.html.
26. D. L. Swofford. PAUP*: Phylogenetic analysis using parsimony (and other methods), 1996. Sinauer Associates, Underland, Massachusetts, Version 4.0.
27. F. Tajima. Evolutionary relationship of DNA sequences in finite populations. *Genetics*, 105:437–460, 1983.
28. N. Takahata. Gene genealogy in three related populations: Consistency probability between gene and population trees. *Genetics*, 122:957–966, 1989.
29. R.A. Welch, V. Burland, G. Plunkett, P. Redford, P. Roesch, D. Rasko, E.L. Buckles, S.R. Liou, A. Boutin, and J. Hackett *et al.* Extensive mosaic structure revealed by the complete genome sequence of uropathogenic Escherichia coli. *Proc. Natl. Acad. Sci. U.S.A.*, 99:17020–17024, 2002.
30. D. Zwickl and D. Hillis. Increased taxon sampling greatly reduces phylogenetic error. *Systematic Biology*, 51(4):588–598, 2002.

Author Index