Teresa M. Przytycka
Marie-France Sagot (Eds.)

LNBI 6833

Algorithms in Bioinformatics

11th International Workshop, WABI 2011
Saarbrücken, Germany, September 2011
Proceedings

Springer

Teresa M. Przytycka   Marie-France Sagot (Eds.)

Algorithms in Bioinformatics

11th International Workshop, WABI 2011
Saarbrücken, Germany, September 5-7, 2011
Proceedings

Springer

Series Editors

Sorin Istrail, Brown University, Providence, RI, USA
Pavel Pevzner, University of California, San Diego, CA, USA
Michael Waterman, University of Southern California, Los Angeles, CA, USA

Volume Editors

Teresa M. Przytycka
National Center for Biotechnology Information
U.S. National Library of Medicine
8600 Rockville Pike, Bethesda, MD 20894, USA
E-mail: przytyck@ncbi.nlm.nih.gov

Marie-France Sagot
Institut National de Recherche en Informatique et en Automatique (INRIA)
and Université Lyon 1 (UCBL)
43 bd du 11 Novembre 1918, 69622 Villeurbanne cedex, France
E-mail: marie-france.sagot@inria.fr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-23037-0 e-ISBN 978-3-642-23038-7
DOI 10.1007/978-3-642-23038-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011934142

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

LNCS Sublibrary: SL 8 – Bioinformatics

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

We are pleased to present the proceedings of the 11th Workshop on Algorithms in Bioinformatics (WABI 2011) which took place in Saarbrücken, Germany, September 5–7, 2011. The WABI 2011 workshop was part of the six ALGO 2011 conference meetings, which, in addition to WABI, included ESA, IPEC, WAOA, ALGOSENSORS, and ATMOS. WABI 2011 was hosted by the Max Planck Institute for Informatics, and sponsored by the European Association for Theoretical Computer Science (EATCS) and the International Society for Computational Biology (ISCB). See https://algo2011.mpi-inf.mpg.de/ for more details. The Workshop in Algorithms in Bioinformatics highlights research in algorithmic work for bioinformatics, computational biology, and systems biology. The emphasis is mainly on discrete algorithms and machine-learning methods that address important problems in molecular biology, that are founded on sound models, that are computationally efficient, and that have been implemented and tested in simulations and on real datasets. The goal is to present recent research results, including significant work-in-progress, and to identify and explore directions of future research.

Original research papers (including significant work-in-progress) or state-of-the-art surveys were solicited for WABI 2011 in all aspects of algorithms in bioinformatics, computational biology, and systems biology. In response to our call, we received 77 submissions for papers and 30 were accepted. In addition, WABI 2011 hosted a distinguished lecture by Vincent Moulton, of the University of East Anglia, UK. We would like to sincerely thank the authors of all submitted papers and the conference participants. We also thank the Program Committee and their sub-referees for their hard work in reviewing and selecting papers for the workshop.

We would especially like to thank Bernard Moret for all his advice and support in carrying out the role of being Co-chairs, as well as EasyChair for making the management of the submissions to WABI such an easy process. Thanks once again to all who participated in making WABI such a success in 2011. For us it has been an exciting and rewarding experience.

June 2011

Teresa Przytycka
Marie-France Sagot

Organization

Program Committee

Additional Reviewers

Degnan, James
Dehof, Anna Katharina
Della Vedova, Gianluca
Dondi, Riccardo
Doyon, Jean-Philippe
Duggal, Geet
Duma, Denisa
Dörr, Daniel
Ehrler, Carsten
El-Kebir, Mohammed
Erten, Sinan
Federico, Maria
Fonseca, Paulo
Francisco, Alexandre
Frid, Yelena
Gambin, Tomasz
Gaspin, Christine
Geraci, Filippo
Ghiurcuta, Cristina
Gorecki, Pawel
Harris, Elena
Hellmuth, Marc
Hoener Zu Siderdissen, Christian
Hormozdiari, Farhad
Huang, Yang
Hufsky, Franziska
Ibragimov, Rashid
Jahn, Katharina
Kelk, Steven
Kennedy, Justin
Kim, Yoo-Ah
Lempel-Musa, Noa
Lin, Yu
Lindsay, James
Linz, Simone
Lorenz, Ronny
Lynce, Ines
Malin, Justin
Manzini, Giovanni
Marschall, Tobias
Menconi, Giulia
Misra, Navodit
Montangero, Manuela
Mozes, Shay
Mueller, Oliver
Nicolae, Marius
Pardi, Fabio
Parrish, Nathaniel
Patro, Rob
Pitkänen, Esa
Polishko, Anton
Roettger, Richard
Rueckert, Ulrich
Ruffalo, Matthew
Russo, Luís
Rybinski, Mikolaj
Salari, Raheleh
Scheubert, Kerstin
Schoenhuth, Alexander
Scornavacca, Celine
Snir, Sagi
Startek, Michal
Stckel, Daniel
Stevens, Kristian
Sun, Peng
Swanson, Lucas
Swenson, Krister
Szczurek, Ewa
Tantipathananandh, Chayant
Taubert, Jan
Thapar, Vishal
Truss, Anke
Ullah, Ikram
Weile, Jochen
Willson, Stephen
Winnenburg, Rainer
Winter, Sascha
Wittkop, Tobias
Wittler, Roland
Wohlers, Inken
Wolter, Katinka
Wu, Taoyang
Wu, Yufeng
Wójtowicz, Damian
Yorukoglu, Deniz
Zaitlen, Noah
Zakov, Shay
Zheng, Yu

Table of Contents

Automated Segmentation of DNA Sequences with Complex Evolutionary Histories

Broňa Brejová, Michal Burger, and Tomáš Vinař

Faculty of Mathematics, Physics, and Informatics, Comenius University,
Mlynská Dolina, 842 48 Bratislava, Slovakia

Abstract. Most algorithms for reconstruction of evolutionary histories involving large-scale events such as duplications, deletions or rearrangements, work on sequences of predetermined markers, for example protein coding genes or other functional elements. However, markers defined in this way ignore information included in non-coding sequences, are prone to errors in annotation, and may even introduce artifacts due to partial gene copies or chimeric genes.

We propose the problem of sequence segmentation where the goal is to automatically select suitable markers based on sequence homology alone. We design an algorithm for this problem which can tolerate certain amount of inaccuracies in the input alignments and still produce segmentation of the sequence to markers with high coverage and accuracy. We test our algorithm on several artificial and real data sets representing complex clusters of segmental duplications. Our software is available at `http://compbio.fmph.uniba.sk/atomizer/`

1 Introduction

Genome rearrangements and segmental duplications, acting on long stretches of DNA, pose a significant challenge to comparative genomics. Rearrangements change the order of segments in the genome, resulting in new gene orders and new chromosomal organization. In a typical rearrangement study, we aim at computing the shortest possible number of operations transforming one genome to another or reconstructing a phylogenetic tree and ancestral gene orders (Moret et al., 2001; Bourque and Pevzner, 2002; Adam and Sankoff, 2008).

Segmental duplications increase the length of the sequences by copying genetic material to new locations, creating complex gene clusters, hotspots of evolutionary innovation (Zhang, 2003). Reconstruction of duplication events within such regions is a key to understanding their organization, function, and evolution (Benson and Dong, 1999; Elemento et al., 2002; Zhang et al., 2009; Vinar et al., 2010; Lajoie et al., 2010).

Most algorithms for these tasks do not work directly on the original sequences, but rather on predetermined markers or synteny blocks. These are intervals of the sequence such that all events in the true evolutionary history introduce breakpoints only at or between the boundaries of these intervals, but not inside them. The notion of such intervals was first introduced by Nadeau and Taylor

(1984) (conserved segments) and in this work we call them *atomic segments* or *atoms*. Splitting the sequence into atoms allows algorithm developers to abstract from modeling local sequence alignments, and instead to concentrate on larger-scale processes that reorder and duplicate blocks of segments.

Here, we introduce a new method for segmenting sequence into atoms, primarily targeted at a fine-scale analysis of relatively recent evolutionary events (for example events that happened in the last 85 My of mammalian evolution, which approximately translates to higher than 80% sequence similarity for neutrally evolving sequences). We test our method in the context of reconstruction of duplication histories; however, it is also applicable to other scenarios, including rearrangement studies. In the rest of this section, we give an overview of the methods used previously for this task, and we demonstrate examples of various problems in real data sets that we address by our work.

Related work. Many algorithms for rearrangement or duplication analysis use protein coding genes as atoms (Fitch, 1977; Benson and Dong, 1999; Moret et al., 2001; Elemento et al., 2002; Bourque and Pevzner, 2002; Bertrand and Gascuel, 2005; Lajoie et al., 2007; Adam and Sankoff, 2008; Lajoie et al., 2010). In order to do so, we need to annotate genes in the sequence and establish homology or even orthology among them. The order and strand orientation of the genes is then used as an input for further analysis.

While this is perhaps the only solution applicable to distantly related sequences that need to be aligned at the protein level, it is not universal. First of all, necessary preprocessing, including gene finding and homology or orthology detection, is difficult and can introduce errors. Even in cases where reliable ortholog sets are available, this approach is not relevant for all evolutionary scenarios. Many incomplete pseudogenes present in human and other genomes clearly show that duplications and rearrangements do not respect gene boundaries. For example, the human PRAME gene cluster contains 38 copies of the PRAME locus (preferentially expressed antigens in melanoma), but more than a third of these copies are incomplete pseudogenes (Gibbs et al., 2007). Even more problematic are *chimeric genes* that contain a breakpoint inside an intron. The PRAME gene cluster contains a chimeric gene whose protein sequence consists of two parts with different phylogenetic ancestries. Its inclusion in a phylogenetic analysis may result in a completely incorrect phylogenetic tree.

Another example, where genes are not appropriate as atoms, is the UGT1A cluster. In the human genome, this cluster contains a single alternatively-spliced gene (UDP-glucuronosyltransferase) with at least 13 unique copies of the first exon that apparently arose by segmental duplication (Bellemare et al., 2010). To analyze this sequence, we would have to use exons as atoms instead of genes, but these exons are too short for a reliable gene tree reconstruction, which is a necessary step for many methods. Therefore it would be ideal to use also some of the surrounding non-coding sequences, which have been copied together with the exons, but that requires finding atomic segments unrelated to functional annotation.

Finally, a well-known KRAB zinc finger gene family in the human genome contains more than 400 genes in 25 gene clusters spread across several chromosomes (Schmidt and Durrett, 2004; Huntley et al., 2006). Each zinc finger gene is composed of the KRAB domain and between three and forty zinc fingers. In this family, we can see duplication of whole genes, as well as duplication of zinc finger domains within genes. For both UGT1A cluster and KRAB genes, the traditional selection of atoms (i.e. first exons or zinc finger domains) requires detailed functional annotation and a complex prior knowledge about the studied region. Even with such knowledge, we may create errors due to chimeric atoms or to loose valuable information due to insufficient sequence coverage.

Recently, a new approach to segmentation based on local sequence alignments has been introduced in the context of ancestral genome reconstruction (Ma et al., 2006). Briefly, using sequence alignment tools such as blastz (Schwartz et al., 2003) or UCSC chain/net pipeline (Kent et al., 2003), they identify significant local alignments of the sequence to itself with a desired level of homology and then use boundaries of these sequence alignments as atomic segment boundaries, and regions between the boundaries as atoms. Ma et al. (2006, 2008b) developed a heuristic pipeline that creates a map of such segments considering events of 50kb or more in length, later refining the boundaries to a finer precision. Their resolution is ideal for mammalian whole-genome analyses; however for smaller-scale events (such as analysis of gene clusters), finer resolution is needed.

In our previous work, we have used a simple greedy heuristic (SGH) for this type of analysis (Vinar et al., 2010). Analyzed sequences are first divided into non-overlapping segments of size 500, and for each segment we search for sequence homologies in the rest of the sequence. The segment with the highest number of homologs and its matching homologs are then designated as atoms. All segments overlapping new atoms are removed and the whole process is repeated.

There are several advantages to methods based on local alignments. The analysis is not dependent on possibly error-prone annotations. The atoms often span longer sections of the sequence which allows more accurate determination of phylogeny of individual atom instances. Finally, thanks to potentially finer resolution of atoms, we can use assumptions of infinite sites model (Ma et al., 2008a), such as low breakpoint reuse assumption that often helps to resolve symmetries in duplication event direction (Zhang et al., 2009).

Problem statement. Our goal is to investigate systematic approaches to segmentation of sequences into atoms. The input for our problem consists of several evolutionarily related sequences or one sequence with segmental duplications. We want to find non-overlapping segments called atoms (completely or partially covering the input sequences) and divide atoms into classes so that: (a) coverage of the sequences by atoms is high, to use as much sequence information as possible, (b) the number of atoms is low, to prevent unnecessary segmentation of long atoms, (c) two atoms of the same class share high sequence similarity across their entire length, (d) two atoms of different classes or two parts of the same atom do not appear to be homologous at a chosen sequence similarity threshold.

In practice, one has to find a trade-off these goals and to accommodate imperfect results of homology detection methods. In the rest of the paper we describe our new algorithm for the segmentation problem and evaluate its performance on both simulated and real sequences.

2 Algorithm

Alignments and breakpoint mapping. The input to our algorithm is a set of evolutionarily related input sequences. We use the LASTZ program (Harris, 2007) to align each sequence to itself and to every other sequence. One possible approach to sequence segmentation is to take the resulting set of local alignments, add a breakpoint at every boundary of a local pairwise alignment and to create an atom between every two adjacent breakpoints (Fig. 1). To classify atoms, we can simply create a graph with atoms as vertices and alignments between atoms as edges. Then we create one class for each connected component of this graph. This approach would work well on a perfect set of alignments, but on real data we can encounter various artifacts.

In Fig. 2, we see an example where one homology was not found by the local alignment program, leading to a missing breakpoint and wrong class assignment in the resulting segmentation. To avoid this problem, we will map boundaries of every alignment through other overlapping alignments to create new breakpoints, as suggested by a dotted line in Fig. 2. Mapping a breakpoint through an alignment is easy if the breakpoint is located at an aligned nucleotide. If it is located at a nucleotide aligned with a gap, we find the nearest aligned nucleotide to the left or to the right (whichever is closer) and map it according to this nucleotide.

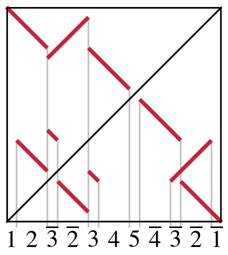


Fig. 1. Simple sequence segmentation. The figure shows a dotplot of alignments of a sequence to itself. We can consider each alignment boundary as a breakpoint; segments between neighbouring breakpoints will form atoms in classes $1, 2, 3, 4, 5$.

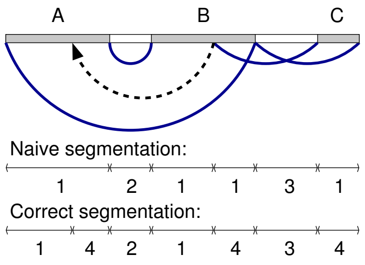
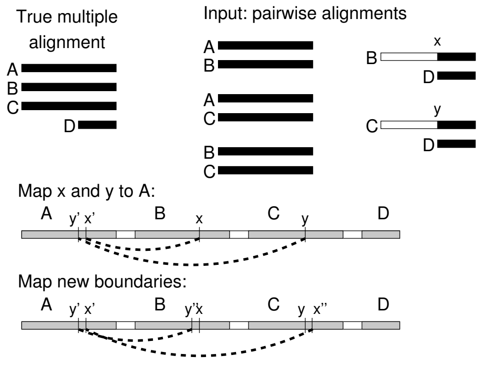


Fig. 2. Example of an input with a missing alignment. Region A is aligned to B and end of B to C, but the alignment between end of A and C is missing. The missing breakpoint in A can be mapped from B through the alignment between A and B, as suggested by the dotted line. Without mapping we get incorrect segmentation.

Iterated homology mapping. In our algorithm we perform breakpoint mapping iteratively, as a newly mapped breakpoint may need to be mapped further to other regions. We call this general process *iterative homology mapping (IHM).*

Another common problem is that boundaries of overlapping local alignments often do not coincide exactly, but are spread around the true breakpoint. This is caused by uncertainty of sequence alignment near alignment boundaries. The problem is even more exacerbated by mapping breakpoints through alignments, as shown in Fig. 3. Breakpoints x and y in regions B and C will be mapped to region A through pairwise alignments. Although ideally they should map to the same place, due to imprecision in alignment boundary between B and D, or due to imprecision in pairwise alignment between A and B, the new breakpoint x' is at some distance from the new breakpoint y'. The new breakpoint x' is then mapped to C and y' is mapped to B, again creating pairs of nearby breakpoints. In some cases, the iterative process may create long arrays of breakpoints originating from repeatedly mapping what should have been the same breakpoints through a cycle of imprecise alignments. It is clearly not desirable to create a very short atom between every pair of such nearby boundaries.



!h

Fig. 3. Example of imprecise breakpoint mapping. Breakpoints x and y are mapped to two different positions in A, and each of them is then further mapped to a new position in B or C. Each of atoms A, B, and C is thus split into three atoms instead of two.

To avoid this problem, we select a window size W and allow at most one breakpoint within each window. This is achieved by clustering breakpoints and replacing each cluster of nearby breakpoints with a new breakpoint roughly at their center. The new breakpoints are chosen so that no two breakpoints are closer than W and the sum of squared distances between the input breakpoints and their new representatives is minimized. This can be done in $O(NW^2)$ time by a dynamic programming algorithm.

Outline of our approach is shown in pseudocode of Algorithm 1. We start with a set of breakpoints created from alignment endpoints. In each iteration of our algorithm, we first cluster breakpoints as described above. Each new breakpoint is then checked against all alignments, and if it is inside an alignment,

Algorithm 1. Iterative homology mapping

```
   Data: set of alignments $A$, window length $W$
1  $B \leftarrow$ endpoints($A$) ;                          // current breakpoints
2  $B_M \leftarrow \emptyset$ ;                              // all mapped breakpoints
3  repeat
4      $B \leftarrow$ cluster($B, W$);
5      $B' \leftarrow$ select_to_map($B, B_M, W$) ;   // get unmapped breakpoints from $B$
6      $B'' \leftarrow$ map($B', A$) ;                     // map $B'$ through $A$
7      $B \leftarrow B \cup B''$;    $B_M \leftarrow B_M \cup B'$;
8  until $B'' = \emptyset$;
9  return $B$;
```

it is mapped to the other sequence in the alignment. However, we do not map breakpoints that were already mapped in one of the previous iterations. Since breakpoint position may change slightly in each iteration due to the clustering process, we only map breakpoints that are at a distance of more than W from every previously mapped breakpoint. For this purpose we keep a list B_M of all previously mapped breakpoints.

During the whole algorithm, we map at most N/W breakpoints, and therefore the algorithm terminates in at most N/W iterations. In practical instances, the number of iterations is usually quite low, ranging from two to six in the experiments reported in this paper.

Atom classification. Once the breakpoints are fixed, we want to group atoms to classes so that the atoms within a class form a cluster densely connected by alignments, and there are relatively few alignments between atoms from different classes. This can be formulated as an optimization problem, where we seek to minimize the weighted sum of the number of false positive alignments (alignments connecting atoms from two different classes) and false negatives (pairs of atoms in the same class not connected by an alignment). This is a weighted variant of the NP-complete Cluster Editing Problem (Shamir et al., 2004).

We solve this problem exactly by CPLEX software from IBM using integer linear programming (ILP) formulation. In order to efficiently process inputs with large numbers of atoms, we employ several simple heuristics. First of all, atoms that are in different components of the alignment connectivity graph will never be in the same class in the optimal solution of the ILP. Therefore we process each connected component separately. We also do not run ILP on components that form a clique, because the optimal solution has then cost 0. If the size of a component exceeds 200, for efficiency reasons we use a different graph clustering strategy implemented in the MCL program (Van Dongen, 2008).

In the process of classification we also assign a strand to each atom so that they are consistent with alignments (each alignment suggests either that the two atoms should be on the same strand or on the opposite strands).

Segmentation postprocessing. In the resulting segmentation, we sometimes see a pair of atom classes a and b such that each atom of class a is always followed by an atom of class b and an atom of class b is always preceded by an atom of

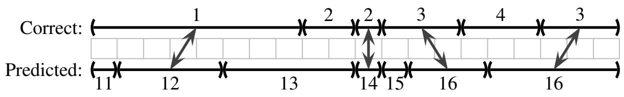


Fig. 4. An example of reciprocal best matches between two segmentations. BRMs are shown as arrows. Each atom is labeled by its class. BRM sensitivity is 4/6, specificity is 4/7. Classes 12 and 16 are correctly predicted.

class a. We replace each such pair by a single atom, because there is no evidence of a breakpoint between a and b at the segmentation level. We perform such postprocessing on predicted segmentations as well as on the true segmentation in the simulated data. If the segmentation does not cover the whole sequence, the two atoms do not need to be adjacent in the sequence, as long as there are no further atoms between them.

Since shorter atoms, with length close to window length W, are often less accurate than longer ones, in some tests we also filter out all classes that have all atoms shorter than some threshold $T > W$.

3 Experiments

Here, we evaluate our methods in the context of duplication history reconstruction of gene clusters from several related species. We have tested our new IHM algorithm on both simulated and real data, comparing it to the simpler algorithm from Vinar et al. (2010), which we will call SGH (Simple Greedy Homology).

Measuring segmentation accuracy. If we know the true segmentation of a sequence, which is the case for artificial sequences generated from an evolutionary model, we can measure the quality of the predicted segmentation directly. To compare two segmentations, we first compute *reciprocal best matches* (BRM) between their atoms (see Fig. 4). In particular, an atom from one segmentation is a BRM of an atom in another segmentation if they cover overlapping regions of the sequence and no other atom overlaps either of the two by a larger amount.

We use four quality measures comparing the predicted segmentation to the true segmentation. Let p the number of atoms in the predicted segmentation, t the number of atoms in the true segmentation, and b be the number of BRM pairs between them. We define BRM sensitivity as b/t and BRM specificity as b/p. If, for example, an algorithm splits a true atom into two predicted atoms, one of them will not have a BRM pair, and therefore the BRM specificity will decrease. Similarly, a predicted atom spanning two real atoms will lead to a decrease in the BRM sensitivity.

The other two measures focus on the correctness of atom classification. Class C_1 is correctly predicted if each of its atoms has a BRM atom in the same true class C_2 and each atom in class C_2 has a BRM atom in C_1. Class sensitivity is then c/t and specificity is c/p, where c is the number of correctly predicted classes, t the number of true classes, and p the number of predicted classes.

Data sets. We have tested our algorithms on 30 simulated data sets divided into three categories with different parameter settings (see the overview in Table 1). To produce them, we have simulated sequence evolution, allowing substitutions according to the HKY model (Hasegawa et al., 1985), short insertions and deletions, as well as large-scale deletions and duplications. The simulation started with a 100kb sequence and proceeded along the human, chimpanzee, and rhesus macaque phylogeny. The parameters of the substitution model, branch lengths of the phylogenetic tree, and rate and length distributions of short insertions and deletions were estimated from UCSC syntenic alignments (Fujita et al., 2011) of human, chimpanzee, and macaque on the human chromosome 22. Parameters of large-scale events (duplications and deletions) were taken from Vinar et al. (2010). The Slow and Fast data sets differ in the rate of large-scale events per site, with Fast data sets using 1.5 times the slower rate. The No indel data sets were taken from Vinar et al. (2010) (first 10 out of 20 sets labeled as 300 in that paper). These sequences were generated by a simpler simulator that did not allow short insertions and deletions and also assumed that the rate of large-scale duplication and deletion is constant per sequence and does not depend on the sequence length.

Segmentation accuracy on simulated data. We have segmented all simulated data sets with our new method IHM, setting $W = 250$ and discarding atoms shorter than 500. The SGH program from Vinar et al. (2010) also produces atoms of length at least 500. On the first two categories, IHM noticeably outperforms SGH, particularly in sensitivity at both the BRM and class levels (Table 2). Since both programs work at the resolution of 500bp, they cannot predict very short atoms, which is why their sensitivity is lower than their specificity. For comparison, we also show the accuracy of the true segmentation with the atoms shorter than 500bp filtered out. Our program is very close to this ideal sensitivity. Even without filtering, IHM maintains high specificity combined with much higher sensitivity than the filtered IHM segmentation.

The last category of data does not contain small-scale indels, which makes the segmentation problem much easier. Due to the higher number of short atoms, the sensitivity is relatively small at resolution of 500bp, but almost identical for both programs as well as for the true segmentation filtered at 500. Both programs have achieved perfect specificity on these data sets, that is, every predicted atom is covered by BRM and all predicted classes agree with the true segmentation. The marked difference is in the boundary placement accuracy. About 86% of

Table 1. Overview of simulated data sets. Each category contains ten data sets. The table lists mean values of various statistics over these data sets, or in the case of sequence length, over all sequences from all data sets in the category combined.

	Sequence	Segmentation		No. events	
Data set	length (kb)	No. atoms	No. classes	dupl.	del.
Slow	253	153	55	28	2.6
Fast	444	385	113	56	3.8
No indels	210	611	78	25	1.2

Table 2. Accuracy of segmentation on simulated sets. TRUE500 is the true segmentation with atoms shorter than 500 filtered out. SGH500 is the segmentation created by method from Vinar et al. (2010). IHM250 is our new algorithm with $W = 250$ and IHM250.500 is the same method, only with atoms shorter than 500 filtered out. Sensitivity and specificity at the BRM and class level are computed as explained in the text.

		BRM		Class	
Set	Program	sn	sp	sn	sp
Slow	TRUE500	86%	100%	89%	100%
	SGH500	63%	97%	45%	83%
	IHM250.500	86%	100%	88%	100%
	IHM250	95%	100%	96%	100%
Fast	TRUE500	80%	100%	86%	100%
	SGH500	65%	99%	52%	91%
	IHM250.500	79%	100%	84%	100%
	IHM250	92%	100%	94%	99%
No indels	TRUE500	55%	100%	61%	100%
	IHM250.500	56%	100%	62%	100%
	SGH500	56%	100%	60%	100%
	IHM250	86%	100%	86%	98%

the BRM atoms produced by IHM have both their boundaries within 50nt of the correct boundary, but this fraction is only 19% for the SGH method. This is because IHM boundaries ultimately originate in alignment endpoints, whereas SGH starts with arbitrarily placed sequence windows.

Influence of segmentation on evolutionary history reconstruction. In Vinar et al. (2010), we have used the true segmentation of simulated sequences as a starting point for evolutionary history reconstruction under a probabilistic model encompassing substitutions and large-scale duplications and deletions. In this work, we compare the accuracy of the history reconstruction when run on different segmentations using a subset of the simulated data from Vinar et al. (2010) (Table 3). We observe that in this case, using the predicted segmentation instead of the true segmentation filtered at 500bp does not have a large impact on the accuracy of the history reconstruction, yielding in most cases the same or very similar number of events.

Segmentation of primate gene clusters. Finally, we have applied our algorithm to the study of three complex primate gene clusters (PRAME, AMY and UGT1A), considered in Vinar et al. (2010). Before running the segmentation algorithms, we have masked the sequences with RepeatMasker, and then excised all masked or unknown bases, obtaining a shorter sequence without repeats.

On all three sets, SGH has produced more atoms (Table 4). A large majority of IHM atoms (81-91% in different sets) have a BRM atom in the SGH segmentation, but only 37%-76% of classes agree completely with the SGH. We have used these new segmentations to infer evolutionary history under the model of large-scale duplications and deletions.

Table 3. The predicted number of evolutionary events by the MCMC method. For each data set D00-D09 in the No indel series, the left column indicates the number of duplications and the right column the number of deletions. The two rows labeled History list the actual number of events in the simulated history, counting only events affecting at least one atom of length of at least 500 or 250 respectively. The remaining rows show difference between the actual count and the count observed in the maximum likelihood history predicted by the MCMC algorithm (Vinar et al., 2010) run on different segmentations.

Program	D00		D01		D02		D03		D04		D05		D06		D07		D08		D09	
History	24	2	24	0	24	1	18	1	24	1	28	1	18	2	22	3	29	1	21	0
TRUE500	0	0	0	0	0	+2	0	0	-1	0	0	0	0	0	-1	+1	-1	1	0	0
SGH500	+1	-1	0	+1	0	+2	0	0	0	0	0	0	0	0	-1	+1	0	0	0	0
IHM250.500	0	0	0	0	+1	+2	0	0	-1	0	0	0	0	0	-1	+1	0	0	0	0
History	25	2	26	0	27	1	18	1	26	1	28	1	19	2	23	3	30	1	22	0
IHM250	0	0	0	+2	+1	+2	0	+1	+1	0	0	+1	0	+1	0	0	-1	+1	0	+1

Table 4. Results on complex primate gene clusters. Data set size, segmentation parameters, and the predicted number of duplications and deletions from the MCMC algorithm (Vinar et al., 2010). Species abbreviations: human (H), chimpanzee (C), orangutan (O), rhesus (R).

		Sequences		Segmentation			No. events	
Set	Program	species	lengths (kb)	atoms	classes	coverage	dup.	del.
PRAME	SGH500	H,R	373,92	454	57	60%	77	21
	IHM250.500			288	51	74%	41	17
AMY	SGH500	H,R	105,89	118	21	82%	14	10
	IHM250.500			90	17	98%	12	8
UGT1A	SGH500	H,C,O	90,87,108	138	17	55%	12	11
	IHM250.500			134	23	76%	11	17

The IHM segmentations consistently lead to fewer events of both kinds (duplications and deletions), with the exception of deletions in the UGT1A cluster. In this case, the IHM segmentation contains four atoms that are unique to the orangutan region. Since the history reconstruction does not allow insertions, these four segments are explained as deletions in the human-chimpanzee lineage. On the other hand, the SGH does not predict atoms with only a single occurrence, and thus these four deletions are not included in the history.

4 Conclusion

We have introduced the problem of automated segmentation of sequences with complex evolutionary histories and proposed an accurate and efficient algorithm. Unlike marker based methods, our algorithm can be used on sequence alone and can be easily incorporated into sequence analysis pipelines. The method is suitable as a preprocessing step for duplication history reconstruction, and it can also be applied to sequences with other large-scale events.

Our method consists of two stages: finding breakpoints and atom classification. Errors introduced in the first stage cannot be resolved later on. One would ideally solve both stages simultaneously. There are two obstacles to this course. First, we need to optimize various contradictory criteria (coverage, the number of false positives and false negatives, etc.). We can either combine them into a single objective function by weights, or we can put a constraint on some criteria by thresholds and optimize the remaining ones, but choosing the weights and thresholds is difficult to do in a principled way. The second problem is that even with breakpoints fixed, the problem of atom classification is already NP-hard (Shamir et al., 2004). Nonetheless, it is possible to investigate heuristics or approximation approaches to tackle this problem.

One can go even further and combine the segmentation and reconstruction of evolutionary histories. Scoring of potential segmentations is then implied directly by the underlying evolutionary model. Such approaches were attempted in the duplication scenario (Zhang et al., 2009; Song et al., 2010). Nonetheless, segmentation makes the history reconstruction problems cleaner and allows us to filter out problematic regions of the sequence (such as shorter atoms in our experiments).

Another, perhaps less ambitious, avenue for improvement, is in the combination of segmentation and multiple alignment. Ideally all atoms of the same class would form a high-quality multiple alignment. Our algorithm relies solely on pairwise alignments, yet multiple alignments of longer homologous regions could help us to map breakpoints more consistently.

Finally, our method was targeted at recently diverged atoms, where it is possible to recognize homology at a nucleotide sequence level. The question of extending our approach to more distant sequences remains open.

Acknowledgements. We would like to thank Webb Miller and Giltae Song for collaboration on various gene cluster problems. This research was supported by Marie Curie grants IRG-224885 and IRG-231025 to TV and BB, and VEGA grant 1/0210/10.

References

Adam, Z., Sankoff, D.: The ABCs of MGR with DCJ. Evolutionary Bioinformatics Online 4, 69–74 (2008)

Bellemare, J., Rouleau, M., Girard, H., Harvey, M., Guillemette, C.: Alternatively spliced products of the UGT1A gene interact with the enzymatically active proteins to inhibit glucuronosyltransferase activity in vitro. Drug Metabolism and Disposition 38(10), 1785–1789 (2010)

Benson, G., Dong, L.: Reconstructing the duplication history of a tandem repeat. In: Intelligent Systems for Molecular Biology (ISMB), pp. 44–53 (1999)

Bertrand, D., Gascuel, O.: Topological rearrangements and local search method for tandem duplication trees. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2(1), 15–28 (2005)

Bourque, G., Pevzner, P.A.: Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Research 12(1), 26–36 (2002)

Elemento, O., Gascuel, O., Lefranc, M.-P.: Reconstructing the duplication history of tandemly repeated genes. Molecular Biology and Evolution 19(3), 278–278 (2002)

Fitch, W.M.: Phylogenies constrained by the crossover process as illustrated by human hemoglobins and a thirteen-cycle, eleven-amino-acid repeat in human apolipoprotein A-I. Genetics 86(3), 623–624 (1977)

Fujita, P.A., et al.: The UCSC Genome Browser database: update 2011. Nucleic Acids Research 39(D), D876–D882 (2011)

Gibbs, R.A., et al.: Evolutionary and biomedical insights from the rhesus macaque genome. Science 316(5822), 222–224 (2007)

Harris, R.: Improved pairwise alignment of genomic DNA. PhD thesis, Pennsylvania State University (2007)

Hasegawa, M., Kishino, H., Yano, T.: Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. Journal of Molecular Evolution 22(2), 160–164 (1985)

Huntley, S., et al.: A comprehensive catalog of human KRAB-associated zinc finger genes: insights into the evolutionary history of a large family of transcriptional repressors. Genome Research 16(5), 669–677 (2006)

Kent, W.J., et al.: Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. Proc. Natl. Acad. Sci. USA 100(20), 11484–11489 (2003)

Lajoie, M., Bertrand, D., El-Mabrouk, N.: Inferring the evolutionary history of gene clusters from phylogenetic and gene order data. Molecular Biology and Evolution 27(4), 761–762 (2010)

Lajoie, M., Bertrand, D., El-Mabrouk, N., Gascuel, O.: Duplication and inversion history of a tandemly repeated genes family. Journal of Computational Biology 14(4), 462–468 (2007)

Ma, J., Ratan, A., Raney, B.J., Suh, B.B., Miller, W., Haussler, D.: The infinite sites model of genome evolution. Proc of the National Academy of Science USA 105(38), 14254–14261 (2008a)

Ma, J., Ratan, A., Raney, B.J., Suh, B.B., Zhang, L., Miller, W., Haussler, D.: DUPCAR: reconstructing contiguous ancestral regions with duplications. Journal of Computational Biology 15(8), 1007–1007 (2008b)

Ma, J., Zhang, L., Suh, B.B., Raney, B.J., Burhans, R.C., Kent, W.J., Blanchette, M., Haussler, D., Miller, W.: Reconstructing contiguous regions of an ancestral genome. Genome Research 16(12), 1557–1565 (2006)

Moret, B.M., Wang, L.S., Warnow, T., Wyman, S.K.: New approaches for reconstructing phylogenies from gene order data. Bioinformatics 17(S1), S165–S173 (2001)

Nadeau, J.H., Taylor, B.A.: Lengths of chromosomal segments conserved since divergence of man and mouse. Proceedings of the National Academy of Science USA 81(3), 814–818 (1984)

Schmidt, D., Durrett, R.: Adaptive evolution drives the diversification of zinc-finger binding domains. Molecular Biology and Evolution 21(12), 2326–2329 (2004)

Schwartz, S., et al.: Human-mouse alignments with BLASTZ. Genome Research 13(1), 103–107 (2003)

Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. Discrete Applied Mathematics 144(1-2), 173–182 (2004)

Song, G., Zhang, L., Vinar, T., Miller, W.: CAGE: combinatorial analysis of gene-cluster evolution. Journal of Computational Biology 17(9), 1227–1232 (2010)

Van Dongen, S.: Graph clustering via a discrete uncoupling process. SIAM Journal on Matrix Analysis and Applications 30, 121 (2008)

Vinar, T., Brejova, B., Song, G., Siepel, A.C.: Reconstructing histories of complex gene clusters on a phylogeny. Journal of Computational Biology 17(9), 1267–1279 (2010)

Zhang, J.: Evolution by gene duplication: an update. Trends in Ecology and Evolution 18(6), 292–298 (2003)

Zhang, Y., Song, G., Vinar, T., Green, E.D., Siepel, A., Miller, W.: Evolutionary history reconstruction for Mammalian complex gene clusters. Journal of Computational Biology 16(8), 1051–1060 (2009)

Towards a Practical $O(n \log n)$ Phylogeny Algorithm

Daniel G. Brown and Jakub Truszkowski

David R. Cheriton School of Computer Science,
University of Waterloo,
Waterloo ON N2L 3G1, Canada
{browndg,jmtruszk}@uwaterloo.ca

Abstract. Recently, we have identified a quartet phylogeny algorithm with $O(n \log n)$ expected runtime, which is asymptotically optimal. Regardless of the true topology, our algorithm has high probability of returning the correct phylogeny when quartet errors are independent and occur with known probability, and when the algorithm uses a guide tree on $O(\log \log n)$ taxa that is correct with high probability. In practice, none of these assumptions is correct: quartet errors are positively correlated and occur with unknown probability, and the guide tree is often error prone. Here, we bring our work out of the purely theoretical setting. We present a variety of extensions which, while only slowing the algorithm down by a constant factor, make its performance nearly comparable to that of neighbour-joining, which requires $O(n^3)$ runtime. Our results suggest a new direction for quartet-based phylogenetic reconstruction that may yield striking speed improvements at minimal accuracy cost.

1 Introduction

Any useful phylogenetic reconstruction algorithm must use $\Omega(n \log n)$ time to reconstruct a phylogeny of n taxa. In practice, all commonly used algorithms take much longer runtimes. To optimize parsimony, likelihood or the least-squares objective is NP-hard, and the best distance-based methods require $\Theta(n^3)$ runtime in the worst case for neighbour-joining, or $\Theta(n^2)$ time for UPGMA [9]. Typical quartet methods require $\Omega(n^4)$ time, since they enumerate all quartets. Common heuristics for the problem run in quadratic time, and the recent program FastTree has a runtime of $\Theta(n^{1.5} \log n)$ in practice. Programs with such a subquadratic runtime are necessary: many projects currently desire trees on hundreds of thousands of taxa, and new projects such as metagenomics or sequencing individual cells inside a human may require creating trees from millions of sequences.

We have recently developed a quite different phylogenetic approach, based on quartet queries, which achieves the $O(n \log n)$ runtime lower bound, in expectation and with high probability, while giving probabilistic guarantees on the quality of its performance in a simple error model; it also promises to return the correct topology with $1 - o(1)$ probability. Our algorithm is randomized, so its runtime is a random variable, but its expectation is $O(n \log n)$ regardless of the true topology of the tree.

However, the probabilistic assumptions on which the program's error analysis depends are not realistic: we require that quartet queries err both independently and with known probability. This presumption is false: for example, two quartet queries that include the same taxon are definitely not independent. Moreover, some quartets may have evidence of being of high quality, while others are not very good. Our initial description of our algorithm did not incorporate this evidence. Our algorithm also only places some taxa into the final tree; with high probability in the theoretical model, all taxa are placed, but in preliminary experiments, 40% of taxa did not find a placement in the tree returned.

Here, we describe our work to bring our algorithm away from a purely theoretical approach to a substantially more practical, and extremely fast, phylogenetic reconstruction method. We describe a number of extensions to our method which increase both the fraction of taxa in the returned tree, which we call *coverage*, and the accuracy of the topology of the tree that is returned. As a result of these extensions, the coverage of our algorithm is now over 90%, and the accuracy of the tree that results approaches that of neighbour joining: see Table 2 in Section 5.

Our methods show that quartet-based algorithms, properly used, can form the basis of an effective phylogenetic reconstruction method, in time that is both theoretically and practically faster than any commonly used approach for the problem.

2 Background and Related Work

Our algorithm is an *incremental* phylogeny algorithm: it starts with a guide tree on a small number of taxa, and then inserts each new taxon s_i into the phylogeny created by the first $i-1$ taxa, until all n taxa are inserted. We use a randomized balanced search tree structure to ensure that, with high probability, it requires $O(\log i)$ time to insert taxon s_i.

2.1 Definitions

A *phylogeny* T is an unrooted binary tree with n leaves in 1-to-1 correspondence with a set $\mathcal{S}$ of terminal taxa. Removing an internal node v from a phylogeny yields three subtrees, $t_i(T, v)$ for $i = 1, 2, 3$. The tree $t_i(T, v)$ joined with its edge to v is the *child subtree* $c_i(T, v)$.

A *quartet* is a phylogeny of four taxa. A *quartet query* $q(a, b, c, d)$, returns one of three possible quartet topologies: $ab|cd$, $ac|bd$ and $ad|bc$: in $ab|cd$, if we remove the internal edge, we disconnect $\{a, b\}$ from $\{c, d\}$. In our work, we assume a quartet query can be done in $O(1)$ time; we discuss how to perform such queries in Section 2.3.

A *node query* $N(T, v, x)$ for internal node v of phylogeny T and new taxon x is a quartet query $q(x, a_1, a_2, a_3)$, where a_i is a leaf of T in $t_i(T, v)$. Such a query identifies the $c_i(T, v)$ where taxon x belongs, if the returned quartet is correct.

2.2 Summary of Our Previous Work

Our algorithm incrementally adds each taxon to the growing tree, using quartet queries to direct the taxon to smaller and smaller parts of the tree. By using a structure that is balanced (with high probability), we ensure that each search requires logarithmic time, giving the asymptotically optimal runtime bound.

We direct the queries using a rooted ternary search tree structure (Figure 1). Each internal node v in our search tree corresponds to a pair: a contiguous region $r(v)$ of the phylogeny, and a node $s(v)$ within that region. Each leaf v of the search tree corresponds to an edge $r(v)$ in the phylogeny. For an internal node v of the search tree, its three children correspond to each of the three $t_i(r(v), s(v))$ subtrees. If node queries are always correct, then asking the node query $(T, s(v), x)$ will identify which of the $t_i(r(v), s(v))$ subtrees the taxon x belongs in, and a traversal through the search tree to a leaf will identify which edge of T is the one where the new taxon x belongs.

Quartet errors are common, which would lead to errors in this simple algorithm. To allow for their existence, we incorporate a random walk method: rather than requiring that we only move down the search tree, we allow for the possibility that we backtrack. Specifically, at node v in the search tree, we ask up to two node queries, corresponding to the boundaries of the current region of the phylogeny; if either of them gives evidence that the new taxon does not belong in $r(v)$, we move to the parent of the current node in the search tree. If they are consistent with $r(v)$, we ask a node query at $(T, s(v), x)$, and that identifies which child of v we go to in the search tree.

Assuming that there is a bias toward going in the correct direction, and that query errors are independent, there is a steady push toward winding up at the correct node in the search tree: the random walk has positive drift, so it moves at an expected constant speed toward the true goal. If we are at a leaf, then we continue the algorithm, but each query that pushes us to stay at the chosen leaf increments a counter and each query that pushes us away from the leaf decrements the counter; we only move up from the leaf in the search tree when the counter has value zero.

This random walk method is inspired by an approach for sorting a list where comparisons are correct with high probability, but err independently with known probability [11]. Our adaptation of the approach increases the number of queries required to insert each new taxon by a constant factor (depending on error probability) over simply descending the search tree. It guarantees that each new taxon is placed in the correct location with $1 - o(1/n)$ probability.

Further, the algorithm runs in $O(n \log n)$ runtime both in expectation and with high probability, since if we use a uniformly-chosen permutation, the search tree remains balanced with high probability. For full details, we refer the reader to our original paper [2].

One key technical detail is that we must ensure that the algorithm never runs out of queries to ask: each new quartet query must be distinct from all the previous ones. This is achieved by first constructing a “guide tree” phylogeny for a large enough subset of taxa. We then build a search tree for the guide tree and

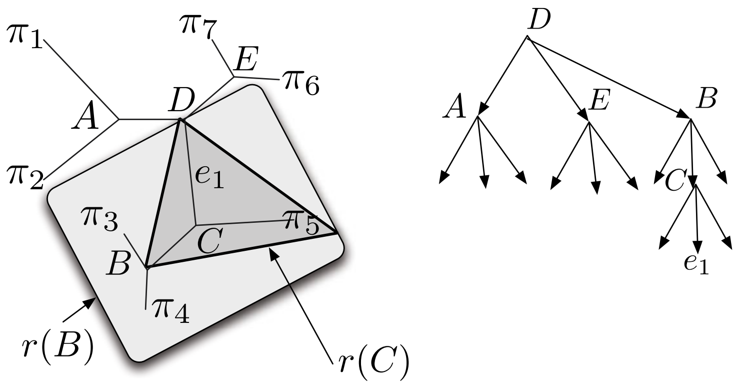


Fig. 1. A phylogeny and a corresponding search tree. Internal nodes of the search tree correspond to subphylogenies: two, for $r(B)$ and $r(C)$, are indicated. Leaves of the search tree correspond to edges of the phylogeny.

insert the remaining taxa into the search tree in random order. Additionally, we guarantee that the guide tree is accurate; in our original paper [2], we show this holds with probability 1-o(1) when the tree is the maximum quartet consistency tree of a set of $\log \log n$ taxa.

2.3 Quartets

There are several ways of inferring the subtopology of a phylogeny corresponding to just four taxa, $\{W, X, Y, Z\}$. Perhaps the most natural is to compute an estimate $d_{i,j}$ for the pairwise distance between all pairs from the four taxa, and then use ordinary least-squares estimation to compute the additive distance matrix closest to the inferred distance matrix, and return the quartet corresponding to the structure of that distance matrix. This is equivalent to comparing $d_{W,X} + d_{Y,Z}$, $d_{W,Y} + d_{X,Z}$ and $d_{W,Z} + d_{X,Y}$; if the smallest of the three values is the first, we infer topology $WX|YZ$, if the second is smallest, we infer $WY|XZ$, and if the third is smallest, we infer $WZ|XY$.

In practice, quartet topology estimation is notably challenging. If the middle edge of the true quartet is short and the other four edges are long, uncertainty in estimating the four pairwise distances that cross the middle edge may dominate the actual length of the edge, making for quartets that are hard to infer. In a balanced phylogeny, a constant fraction of all quartets will have this problem.

As the traversal of the search tree progresses, the region of the tree that is under search shrinks, so quartets will correspond to smaller total lengths, meaning that the fraction of the total edge weight of the quartet that is falling on the middle edge will drop as well; this suggests that quartet errors may be rarer at lower levels of the search tree, as we discuss in Section 3.

Other methods to infer quartets include the ordinal quartet method [12], weighted least-squares, and using maximum-likelihood methods. We note that our algorithm assumes all of these methods operate in $O(1)$ time on a quartet, as our runtime does not depend on the phylogenetic source data, whether it is biological sequences, an alignment of such sequences, or microsatellite or other data with phylogenetic signal in it. Indeed, our algorithm works in any scenario in which quartets can be successful with sufficiently high probability.

2.4 Other Fast Phylogenetic Algorithms

The tree that optimizes the neighbour-joining objective is not known to be producible in $o(n^3)$ time in worst case. However, heuristics for this problem have been a subject of much research; Wheeler [20] gives an exact neighbour-joining implementation, using clever data structures, which seems to require sub-cubic runtime in typical instances, while other authors [7,14] have given heuristic algorithms that operate in $O(n^2)$ time and that typically run in $O(n\sqrt{n}\log n)$ time, but do not guarantee to optimize the neighbour-joining objective. The UPGMA objective can, be optimized in $O(n^2)$ runtime [9]. Desper and Gascuel [6] give a heuristic for the minimum-evolution objective that runs in $O(n^2 \log n)$ for most trees.

Erdös and co-authors have shown that the short quartet method can solve the phylogenetic inference problem in $O(n^2 \text{polylog } n)$ time on most trees, on sufficiently long sequences, and assuming standard Markov models of evolution [8]; Csűros gives a practical algorithm that has similar guarantees [4]. There is also a developing literature on sequence length requirements for various phylogeny algorithms, and on identifying parts of the tree that can be reconstructed from a given alignment [5].

Sub-quadratic phylogeny methods are quite rare: King *et al.* [13] give an $O(n^2 \frac{\log\log n}{\log n})$-time algorithm that is not widely used. When data are error free, our algorithm is not the first $O(n \log n)$ algorithm, as Kannan *et al.* [10] have given a rooted-triple algorithm with this runtime.

A more distressing lower bound, also in the same paper of King *et al.*, shows that any distance-based method requires $\Omega(\frac{n^2}{\log n})$ time to reconstruct correct trees where simple Markov evolution models are used, and distances are inferred using standard techniques, assuming sequences are quite short.

2.5 Quality Measures

We present two different quality measures: Robinson-Foulds quality and quartet quality. Given a ground-truth topology for a set of taxa and a second topology on the same taxa, the Robinson-Foulds quality is the fraction of the non-trivial splits (internal edges) of one topology found in the other. The quartet quality (see e.g. [3,1]) is the fraction of the $\binom{n}{4}$ quartets that have the same result in both tree topologies. We note that the Robinson-Foulds measure is fragile: a single misplaced taxon can ruin all of the splits of the tree. By contrast, the quartet distance is robust to individual errors: even a randomly-placed taxon will probably have the correct topology in one-third of its quartets, as there are only three choices for the topology.

3 Extensions to Improve Performance

The basic algorithm suffers from a number of issues that greatly limit its practical performance. The accuracy of quartet inference is often very low, particularly for quartets asked at the top nodes of the search tree; when running the basic algorithm on the COG840 data set, we found that only 66% of quartets were inferred correctly, with only 50% of correct inferences at the root of the search tree. In practice, quartet inference errors are also not independent, which may cause the random walk to "drift away" from the optimal placement of the taxon.

3.1 Quartet Weights

The basic algorithm considers all quartets to be equally reliable. In practice, some quartets are more likely to be correct than others. Assigning equal weight to all quartets can lead to serious errors due to many erroneous quartets. Various quartet phylogeny algorithms estimate the reliability of inferred quartet topologies based on their likelihood scores [19,18,15] or distances between taxa (e.g. [16]).

We assign quartet weights based on the inferred middle edge length normalized by the the sum of all the edge lengths of the quartet. For a quartet with external edge lengths a, b, c, d and an internal edge length e, this becomes $w = e/(a+b+c+d+e)$. The intuition behind this idea is that if the inferred middle edge is long compared to the distances between taxa, then the inferred quartet topology is less likely to have arisen from errors in the distance estimates. The edge lengths are inferred using ordinary least squares, which was dictated by the speed of this approach. This contrasts with most quartet algorithms, where quartets are inferred using maximum likelihood.

We note that this weighing scheme is somewhat different than other commonly used schemes. Most quartet algorithms use likelihood weights as opposed to distances. Rao *et al.* use the topological diameter of the quartet in a preliminary tree constructed by Neighbor Joining.

To incorporate the reliability information into the algorithm, we ask multiple quartet queries at each node query and vote according to the weights. We have tested two voting schemes. In the weighted-majority scheme, the weights of each quartet pointing in the same direction are added and the direction with the highest vote total is chosen. In the winner-takes-all scheme, the direction is chosen according to the quartet query with the highest weight. We found that the weighted-majority voting scheme results in better accuracy according to the quartet measure, while the winner-takes-all approach yields higher accuracy according to the Robinson-Foulds measure (see Section 5).

3.2 Biased Choice of Quartets

Many authors have suggested that large distances between taxa lower the accuracy of quartet inference [16,8]. While our algorithm is forced to ask long quartets at the top of the search tree, biasing the choice towards shorter quartets at the lower nodes of the search tree might reduce the number of unreliable quartets. When asking a node query at node y of the search tree, we require that the representatives for each quartet query are contained in a subtree $r(y^*)$ associated with a node y^* that is at most d levels above y in the search tree. The value d is chosen for each y so that the number of possible representatives in each direction is at least 20. Note that this does not change the behaviour of the algorithm in the upper parts of the search tree, since the subtrees associated with the upper nodes are typically much larger.

3.3 Multiple Insertion Rounds

Due to the ambiguity in inferred quartets, the random walk will often terminate at an internal node in the search tree, resulting in the taxon not being inserted into the phylogeny. After the algorithm has attempted to insert every taxon in the search tree, we try reinserting the taxa that did not make it to a leaf in the first round. We have two such rounds of reinsertions.

3.4 Confidence Threshold

Incorrectly inserting a taxon can prevent subsequent taxa from being inserted in the correct place. To mitigate this, we only add a new taxon to the phylogeny if it has spent more than k last steps of the random walk at the current leaf.

3.5 Repeating the Random Walk

To improve our confidence in the placement of new taxa, we ran the random walk two times for each inserted taxon. The taxon was then inserted only if both walks terminated at the same leaf node.

4 As-Yet Unsuccessful Ideas

Several natural methods for improving our algorithm's performance have not yet been successful. We note these extensions here to document the challenging process of improving a theoretical prototype into a useful method for phylogenetic inference.

Increasing the number of quartet queries without weighting them improved the fraction of taxa inserted, but decreased the accuracy. This emphasizes the need for weighing quartets.

We have tried various other methods of estimating the reliability of quartets. Earlier work on short quartet methods [16,8] suggested using the diameter of the quartet for estimating its reliability. Contrary to our expectations, this approach did not provide satisfactory results. This may be because the diameter of the inferred quartets varies widely between different levels of the search tree. Quartets inferred near the root of the search tree tend to have large distances between taxa, whereas quartets inferred in the deeper parts of the search tree are shorter since they only span a small fraction of the overall tree. Using least squares fit of the distances to the quartet topology also did not seem to improve the accuracy of node queries, and using weighted least squares also did not improve the accuracy of quartet inferences.

We have tried to find a good starting point for the random walk by using profile search. For each of the top $\log n$ nodes of the search tree, we constructed a profile of sequences in the subtree associated with that search tree node. Before starting the random walk for the new taxon, we matched its sequence to the profiles and start the random walk from the highest-scoring subtree. This did not have a significant effect on the results, perhaps because of the large diversity of sequences within the top subtrees.

5 Experiments

We have evaluated our algorithm on several simulated data sets. The simulated data sets are taken from Price *et al.* [14] who used them to evaluate their heuristic program FastTree, which has $O(n^{1.5} \log n)$ runtime on typical instances. These data sets were generated by taking real protein alignments and their maximum likelihood trees, and then simulating evolution of sites along the phylogenetic tree. The evolutionary rates varied across sites.

In most experiments, the initial guide tree was created using Neighbor Joining on a randomly chosen subset of 200 taxa. We used ScoreDist [17] to estimate distances for all versions of our algorithm. FastTree uses a similar, but distinct, distance measure [14]. For a fair comparison of accuracies and running times, we only compare against the initial Neighbor Joining phase of FastTree. FastTree then performs nearest neighbour interchanges to improve the quality of the resultant tree, which takes up much of its runtime; we have not incorporated these into our algorithm, either, so we compare against the initial phase of FastTree for consistency of comparison.

In the first experiment, we assessed how various improvements discussed in Section 3 changed the performance of the algorithm. We ran each version of the algorithm 100 times on the COG840 data set. The results are given in Table 1.

Overall, the improvements to the algorithm boosted the RF accuracy from 46% to around 60%, and increased the proportion of inserted taxa from 56% to 82-95%, depending on the settings. The biggest gains were achieved by introducing quartet weights and a confidence threshold. Multiple rounds of insertions increased the coverage of the taxa set. Increasing the number of quartets and running the random walk twice increased the accuracy, but at the cost of increased running time. It should be noted, however, that the increase in running time was not directly proportional to the number of quartets per query: one run of the algorithm for 5 quartets per query takes around 11 seconds, compared to just under 20 seconds for 20 quartets per query. This is because the running time is dominated by computing the distances, which we save and reuse in the subsequent quartet queries.

Table 1. Shown are results for the COG840 data set with 1250 taxa. We show our algorithm's performance in various settings, and compare it to Neighbor Joining. We report accuracies using the Robertson-Foulds measure. Our algorithm places approximately 80% − 90% of taxa with accuracy around 60%. We ran each version of the algorithm 100 times. In all cases, the guide tree is on 200 taxa; except in the second line of the table, this was generated with Neighbor Joining, and had RF accuracy of $50\% \pm 3\%$.

method	taxa inserted	% taxa inserted	overall accuracy
basic RW+NJ guide tree	704 ± 31	56.3 ± 2.4	46.3 ± 4.6
basic RW+true guide tree (not feasible in practice)	716 ± 26	57.3 ± 2.1	49.4 ± 5.0
5 quartets per node query	955 ± 25	76.4 ± 2.0	41.0 ± 3.8
5 quartets, weights	1070 ± 21	85.6 ± 1.6	48.6 ± 3.7
5 quartets, weights, 2 extra rounds	1205 ± 13	95.4 ± 1.0	45.5 ± 3.4
confidence threshold	1024 ± 19	81.9 ± 1.5	58.4 ± 4.0
re-running the RW	985 ± 30	78.8 ± 2.4	59.5 ± 3.7
WTA voting, no re-run	1003 ± 26	80.2 ± 2.1	62.3 ± 2.9
WTA voting, 20 quartets per query	1151 ± 18	92.1 ± 1.4	60.8 ± 2.9
NJ	1250	n/a	62.6

In general, WTA voting tended to produce trees with higher RF accuracy than weighted-majority voting. In contrast, weighted-majority voting resulted in higher quartet accuracy, as we shall see in the next experiment.

We see that there is a trade-off between the proportion of the inserted taxa and the RF accuracy. This is not surprising since incorrectly inserting a new taxon into the phylogeny can cause many splits to be incorrect.

The accuracy of guide trees was considerably lower than that of the NJ tree for the full data set. To investigate the impact of errors in the guide tree on the accuracy of the algorithm, we ran the algorithm starting from a guide tree consistent with the true phylogeny. Contrary to our expectations, errors in the guide tree have little impact on the accuracy; substituting the NJ guide tree with the true one improved the accuracy by 2 to 3 percent, depending on the version of the algorithm.

In the second experiment, we evaluated the algorithm on three unrelated data sets having 250, 1250, and 5000 taxa, respectively. For each data set, we ran the algorithm in four different configurations, using two voting schemes and varying the number of quartet queries per node query. The size of the guide tree was 200 except for the 250 taxon data set, where it was set to 100. The results are shown in Table 2.

We see that the weighted-majority voting scheme tends to produce trees with higher quartet accuracy, whereas the winner-takes-all voting scheme yields higher RF accuracy. In general, we see that the two measures are very different: an algorithm can have relatively good performance according to one while being considerably worse in the other. In most cases, the accuracy of the random walk algorithm is lower than the accuracy of Neighbor Joining.

Table 2. Performance of the random walk algorithm on synthetic alignments.The size of the guide tree was 200 except for the 250 taxon data set, where it was set to 100. The average RF accuracy of the guide trees was 65, 50, and 46% for the 250, 1250, and 5000-taxon data sets, respectively. The average quartet accuracies for the guide trees were 73, 83 and 55%.

	data set								
	COG1011 250 sequences			COG840 1250 sequences			COG1028 5000 sequences		
method	% taxa	RF	QA	% taxa	RF	QA	% taxa	RF	QA
weighted majority,5 quartets	88.8	67.0	72.8	83.6	58.6	85.8	73.3	51.9	59.6
WTA-vote,5 quartets	86.0	69.4	70.8	80.0	62.3	85.4	70.1	57.0	59.6
weighted majority,20 quartets	95.6	60.4	69.9	96.4	50.6	83.9	94.3	41.1	56.5
WTA-vote,20 quartets	94.0	69.4	73.4	92.1	60.8	83.1	89.7	57.6	55.3
NJ	100	73.6	70.0	100	62.6	88.0	100	73.0	66.3
FastTree(NJ phase only)	100	69.7	85.9	100	61.0	86.6	100	73.6	66.4

In the final experiment, we measured the running time of our algorithm on large data sets. For this experiment, we took a large simulated nucleotide alignment based on a real 16S alignment and compared the runtimes of the initial phase of FastTree and our algorithm. We ran the algorithms on the full alignment and on two smaller alignments created by randomly sampling 20000 and 40000 sequences from the large alignment. All running times were measured on a standard desktop computer with an AMD 7750 Dual Core 2712MHz processor and 4 GB RAM. In all cases, our algorithm runs faster, with the relative speedup increasing with the size of the data set; Table 3 shows the results.

Table 3. Running times of the random walk algorithm compared to FastTree. We used the huge.1 alignment from the original FastTree paper [14]. Smaller data sets were created by choosing a random subset of sequences from the large alignment. Our algorithm runs 2.1 to 3.4 times faster than FastTree on these very large data sets.

	# of sequences		
	20,000	40,000	78,132
weighted majority, 5 quartets	6m 41s	15m 52s	34m
FastTree (NJ phase only)	13m 52s	41m 15s	116m

6 Conclusion

We have presented our work in progress to move our extremely fast quartet phylogeny algorithm from being a theoretical result to a practical algorithm for inferring trees. We have shown that a variety of sensible heuristics, including weighting quartets by our confidence in them, using multiple quartet queries per insertion, and using multiple rounds of insertion, can increase both our coverage and our accuracy substantially over our original implementation. At present, our algorithm is close to being competitive with neighbour-joining, while being much faster. Certainly, for very large data sets, it may be suggested as one of few heuristics for the task.

There is still much future work: in particular, we are currently limited in part due to bad quartet inferences. If there were a way to bias our quartet choice toward high-quality quartets, this could substantially improve our quality. Also, it would be helpful if, after each round of insertions of taxa, we could remove taxa that are likely in a poor placement, and re-insert them. Finally, we are investigating a profiling technique, as described in Section 4, so as to pre-identify a good starting search node for each new taxon, instead of the root, where we predict errors are more common.

It remains to be seen whether a very fast quartet-based phylogeny method of our sort can in fact be competitive with the best algorithms. However, over the years, a number of other methods have grown up trading off speed with accuracy. We see evidence that that speed can be pushed to the theoretical limit, and that good trees can result from such procedures.

References

1. Brodal, G.S., Fagerberg, R., Pedersen, C.N.S.: Computing the quartet distance between evolutionary trees in time O(n log n). Algorithmica 38(2), 377–395 (2003)
2. Brown, D.G., Truszkowski, J.: Fast error-tolerant quartet phylogeny algorithms. In: Giancarlo, R., Manzini, G. (eds.) CPM 2011. LNCS, vol. 6661, pp. 147–161. Springer, Heidelberg (2011)
3. Bryant, D., Tsang, J., Kearney, P.E., Li, M.: Computing the quartet distance between evolutionary trees. In: Proceedings of SODA 2000, pp. 285–286 (2000)
4. Csűrös, M.: Fast recovery of evolutionary trees with thousands of nodes. J. Comp. Biol. 9(2), 277–297 (2002)
5. Daskalakis, C., Mossel, E., Roch, S.: Phylogenies without branch bounds: Contracting the short, pruning the deep. In: Batzoglou, S. (ed.) RECOMB 2009. LNCS, vol. 5541, pp. 451–465. Springer, Heidelberg (2009)
6. Desper, R., Gascuel, O.: Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. J. Comp. Biol. 9(5), 687–706 (2002)
7. Elias, I., Lagergren, J.: Fast neighbor joining. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1263–1274. Springer, Heidelberg (2005)
8. Erdös, P.L., Steel, M.A., Székely, L.A., Warnow, T.: A few logs suffice to build (almost) all trees: Part II. Theor. Comput. Sci. 221(1-2), 77–118 (1999)
9. Gronau, I., Moran, S.: Optimal implementations of UPGMA and other common clustering algorithms. Inf. Process. Lett. 104(6), 205–210 (2007)
10. Kannan, S.K., Lawler, E.L., Warnow, T.J.: Determining the evolutionary tree using experiments. J. Algorithms 21(1), 26–50 (1996)
11. Karp, R.M., Kleinberg, R.: Noisy binary search and its applications. In: Proceedings of SODA 2007, pp. 881–890 (2007)
12. Kearney, P.E.: The ordinal quartet method. In: Proceedings of RECOMB 1998, pp. 125–134 (1998)
13. King, V., Zhang, L., Zhou, Y.: On the complexity of distance-based evolutionary tree reconstruction. In: Proceedings of SODA 2003, pp. 444–453 (2003)

14. Price, M.N., Dehal, P.S., Arkin, A.P.: FastTree: Computing large minimum evolution trees with profiles instead of a distance matrix. Mol. Biol. Evol. 26(7), 1641–1650 (2009)
15. Ranwez, V., Gascuel, O.: Quartet-based phylogenetic inference: Improvements and limits. Mol. Biol. Evol. 18(6), 1103–1116 (2001)
16. Snir, S., Warnow, T., Rao, S.: Short quartet puzzling: A new quartet-based phylogeny reconstruction algorithm. J. Comp. Biol. 15(1), 91–103 (2008)
17. Sonnhammer, E.L.L., Hollich, V.: Scoredist: A simple and robust protein sequence distance estimator. BMC Bioinf. 6, 108 (2005)
18. Strimmer, K., Goldman, N., von Haeseler, A.: Bayesian probabilities and quartet puzzling. Mol. Biol. Evol. 14(2), 210–211 (1996)
19. Strimmer, K., von Haeseler, A.: Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. Mol. Biol. Evol. 13(7), 964–969 (1996)
20. Wheeler, T.J.: Large-scale neighbor-joining with NINJA. In: Salzberg, S.L., Warnow, T. (eds.) WABI 2009. LNCS, vol. 5724, pp. 375–389. Springer, Heidelberg (2009)

A Mathematical Programming Approach to Marker-Assisted Gene Pyramiding

Stefan Canzar[1,⋆] and Mohammed El-Kebir[1,2,⋆]

[1] Centrum Wiskunde & Informatica, Life Sciences Group,
Science Park 123, 1098 XG Amsterdam, The Netherlands
{s.canzar,m.el-kebir}@cwi.nl

[2] Centre for Integrative Bioinformatics VU (IBIVU), VU University Amsterdam,
De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands

Abstract. In the *crossing schedule* optimization problem we are given an initial set of parental genotypes and a desired genotype, the ideotype. The task is to schedule crossings of individuals such that the number of generations, the number of crossings, and the required populations size are minimized. We present for the first time a mathematical model for the general problem variant and show that the problem is $\mathcal{NP}$-hard and even hard to approximate. On the positive side, we present a mixed integer programming formulation that exploits the intrinsic combinatorial structure of the problem. We are able to solve a real-world instance to provable optimality in less than 2 seconds, which was not possible with earlier methods.

1 Introduction

Plant breeding is the practice of creating improved varieties of cultivated crops with for instance a higher yield, better appearance or enhanced disease resistance [2]. Up to recently, selection of favorable traits has been solely on the basis of observable *phenotype* [4]. With the availability of *genetic maps*, containing the exact locations on the genome of genetic markers associated with desirable traits, selection at the *genotypic* level has become possible [8]. This knowledge allows to design a schedule of crossings of individuals resulting ultimately in an individual with all alleles corresponding to desired favorable traits present. In the plant breeding literature this process is called *marker-assisted gene-pyramiding* and the resulting plan a *gene-pyramiding scheme* or a *crossing schedule* [3, 10, 14]. In this work we consider a mathematical programming approach to the problem that asks to identify given (1) a genetic map, (2) an initial set of parental genotypes and (3) the desired genotype—the so called *ideotype*—a crossing schedule that results most cost-efficiently in the ideotype with respect to the following three criteria. Firstly, it takes time for the progeny to mature such that a next crossing can be performed. So the *number of generations* is a measure on the time it takes to execute the crossing schedule. Secondly, every crossing between

⋆ Joint first authorship.

two individual plants requires an effort from the breeder, e.g. plants have to be treated such that they flower at the same time. So typically the *number of crossings* is also to be minimized. Thirdly, in order to obtain the genotypes required by the schedule, for every crossing a specific number of offspring need to be generated among which the desired genotype is expected to be present. Simply speaking, the more difficult it is to obtain the desired genotype out of its parental genotypes, the larger the required number of offspring will be. Since every individual in the offspring has to be screened for having the desired genotype, the *total population size* is also to be minimized.

Related work. Most work on gene pyramiding lacks a formal framework; instead only an overview of guidelines and rules of thumb is given [6, 14]. A notable exception, however, is the work by Servin et al. [10] who were the first to introduce a special case of the problem considered in this paper in a formal way. The authors show how to make use of the genetic map in determining the population sizes needed for all crossings. Contrary to our formulation, they allow a genotype to only participate in one crossing. In addition, very restrictive assumptions about the genotypes of the initial parents were made. These restrictions allowed the authors to exhaustively enumerate all crossing schedules and compare them in terms of population size needed. By introducing a heuristic, which partially alleviates the restriction on re-use of genotypes, the authors could compute smaller population sizes for the instances considered. Later papers by Ishii and Yonezawa [6] assume that target genes are always unlinked, which imposes a lower bound on the genetic distance of pairs of target genes. Similar to our work, in [6] the number of generations, number of crossings and the total population size are identified as important attributes. An experimental evaluation is performed on manually obtained crossing schedules having different topologies for a fixed number of parents.

Our contribution. In this work we lift the restrictions imposed by Servin et al. and consider a more general variant of the problem where genotypes are allowed to be re-used and no assumption about the initial parental genotypes is made. For the first time we formulate a mathematical model of the general problem. We show NP-hardness using an approximation-factor preserving reduction from an inapproximability result follows. We introduce a mixed integer linear program (MIP) formulation which exploits various aspects of the inherent combinatorial structure of the problem and which approximates the non-linear objective by a piecewise linear curve. Finally, we show that our approach is capable of solving real-world instances to provable optimality within a precise mathematical model, which was not possible with earlier methods. The rest of the paper is organized as follows. We start by formally defining the problem and showing hardness of the problem. In Section 3 we introduce our method and state a MIP formulation. An experimental evaluation on a real-word instance and on randomly generated instances is presented in Section 4. We conclude with a discussion on our results in Section 5. Due to the lack of space, we omit the proofs of the given lemmas.

2 Problem Definition and Complexity

A *genotype* C is a $2 \times m$ matrix whose elements are called *alleles*. The two rows, $C_{1,\cdot}$ and $C_{2,\cdot}$, are called the lower and upper *chromosome*, respectively. Each column in C corresponds to a *locus*. So at a locus p two alleles are present, which we denote by $c_{1,p}$ and $c_{2,p}$. A locus is said to be *homozygous* if its two alleles are identical, otherwise it is *heterozygous*. Likewise, a genotype is homozygous if all its loci are homozygous, otherwise the genotype is said to be heterozygous. The desired genotype is called the *ideotype*, which we denote by C^*. In plant breeding often pure lines are desired, as they allow for instance for the production of F1 hybrids [2]. Therefore for the remainder of the paper we assume the ideotype to be homozygous. In this case, actual alleles can be classified as being present in the ideotype or not. Hence, the alleles in any genotype C are binary.

We represent a *crossing schedule* as a *connected directed acyclic graph* (DAG) whose nodes are labeled by genotypes. Specifically, the source nodes correspond to the initial parental genotypes. A non-source node, which we refer to as an *inner node*, corresponds to a crossing. The single target node is labeled by the ideotype. The arcs are directed towards the ideotype and relate a parent with its child. Since a genotype is obtained from two parents, the in-degree of an inner node is exactly 2. The two parents of a node need not be distinct. We say that a genotype is obtained via *selfing* if its two parents are identical. From the topology of a crossing schedule the number of generations and the number of crossings can be inferred. The number of generations is the length of the longest path from a source node to the target node. On the other hand, the number of crossings corresponds to the number of inner nodes. In Figure 1 an example crossing schedule is given.

The third attribute of a crossing schedule, the *total population size*, is the sum of the population sizes implied by the crossings represented by inner nodes. Let C be the genotype of an inner node and let D and E be the genotypes of the two parents of C. Later, we will show what the probability $\Pr[\, D, E \to C \,]$ of obtaining C out of D and E is. For now we denote this probability with ρ. The population size $N(\rho, \gamma)$ corresponding to ρ is the number of offspring one needs to generate in order to find with a given *probability of success* γ an individual with genotype C among the offspring. Since ρ is the probability of success in a Bernoulli trial, the probability that none of the $N(\rho, \gamma)$ offspring have genotype C is $(1 - \rho)^{N(\rho,\gamma)} = 1 - \gamma$. Therefore we have that

$$N(\rho, \gamma) = \frac{\log(1 - \gamma)}{\log(1 - \rho)}. \tag{1}$$

As also remarked in [10], it is sensible to have an upper bound on every population size in the schedule, as depending on the plant species only a limited number of offspring can be generated. For that purpose we define $N_{\max}$ to be the *maximal population size* to which every crossing in a crossing schedule has to adhere.

In diploid organisms, the genotype of a zygote is obtained by the fusion of two haploid gametes originating from one parent each. So one of the chromosomes of the resulting genotype C, say $C_{1,\cdot}$, corresponds to a gamete given rise to by D and the other chromosome corresponds to a gamete produced by E. A gamete is the result of a biological process called *meiosis* where in pairs of homologous chromosomes crossover events may occur. In our setting, this means that an allele $c_{1,p}$ corresponds to either $d_{1,p}$ or $d_{2,p}$ (where $1 \leq p \leq m$). In case a pair of alleles at loci p and q of $C_{1,\cdot}$ do not correspond to the same chromosome of D, we say that a *crossover* has occurred between loci p and q (see Figure 1). From the genetic map, the probability of having a crossover between any pair of loci can be inferred using for instance Haldane's mapping function [5]. Let R be a $m \times m$ matrix containing all crossover probabilities. Due to the nature of meiosis, we have that $r_{p,q} \leq 0.5$ for $1 \leq p < q \leq m$. Let $s = (\nu(1), \ldots, \nu(k))$ be an ordered sequence of heterozygous loci in D. The probability of obtaining $C_{1,\cdot}$ out of D, i.e. $\Pr[\,D \to C_{1,\cdot}\,]$, is then as follows [10]. If there is an allele in $C_{1,\cdot}$ that does not occur in D at the same locus then $\Pr[\,D \to C_{1,\cdot}\,] = 0$. Otherwise, if s is empty then $\Pr[\,D \to C_{1,\cdot}\,] = 1$. Otherwise

$$\Pr[\,D \to C_{1,\cdot}\,] = \frac{1}{2}\prod_{i=1}^{k-1}\begin{cases} r_{\nu(i),\nu(i+1)} & \text{if } c_{1,\nu(i)} = d_{1,\nu(i)} \wedge c_{1,\nu(i+1)} = d_{2,\nu(i+1)} \\ & \text{or } c_{1,\nu(i)} = d_{2,\nu(i)} \wedge c_{1,\nu(i+1)} = d_{1,\nu(i+1)} \\ 1 - r_{\nu(i),\nu(i+1)} & \text{otherwise.} \end{cases} \tag{2}$$

We can now compute $\Pr[\,D, E \to C\,]$ using the following lemma.

Lemma 1. *The probability of obtaining C out of genotypes D and E is*

$$\Pr[\,D, E \to C\,] = \begin{cases} \Pr[\,D \to C_{1,\cdot}\,] \cdot \Pr[\,E \to C_{2,\cdot}\,] & \textit{if } C_{1,\cdot} = C_{2,\cdot} \\ \Pr[\,D \to C_{1,\cdot}\,] \cdot \Pr[\,E \to C_{2,\cdot}\,] & \\ \qquad + \Pr[\,E \to C_{1,\cdot}\,] \cdot \Pr[\,D \to C_{2,\cdot}\,] & \textit{if } C_{1,\cdot} \neq C_{2,\cdot} \end{cases} \tag{3}$$

A common way to deal with multiple objectives is to consider a convex combination of the objective criteria involved [12]. Given a crossing schedule, let crs, gen and pop denote the number of crossings, number of generations and the total population size, respectively. For $\lambda_{\text{crs}}, \lambda_{\text{gen}}, \lambda_{\text{pop}} \geq 0$ and $\lambda_{\text{crs}} + \lambda_{\text{gen}} + \lambda_{\text{pop}} = 1$, the *cost* of that crossing schedule is given by the convex combination $\lambda_{\text{crs}} \cdot \text{crs} + \lambda_{\text{gen}} \cdot \text{gen} + \lambda_{\text{pop}} \cdot \text{pop}$.

*Problem 1 (*CrossingSchedule*).* Given $\mathcal{P} = \{C^1, \ldots, C^n\}$, the set of parental genotypes we start with, the homozygous ideotype $C^* \notin \mathcal{P}$, the recombination matrix R, the desired probability of success $\gamma \in (0, 1)$, the maximal population size $N_{\max} \in \mathbb{N}$ allowed per crossing, and a vector λ of the cost efficients, problem CrossingSchedule asks for a crossing schedule of minimum cost.

We propose a polynomial-time reduction from the decision problem SetCover [7]: the loci correspond to the elements in the universe and the initial set of parents to the family of subsets. The first chromosome of a parent C^i has a 1 at locus p if p is contained in the corresponding subset. The second chromosomes of all parental genotypes consists of only zeros. The ideotype has 1 alleles at every locus. In the cost function we only consider the number of crossings, i.e. $\lambda_{\text{crs}} = 1$ and $\lambda_{\text{gen}} = \lambda_{\text{pop}} = 0$.

Theorem 1. CROSSINGSCHEDULE *is NP-hard.*

Due to the approximation-factor preserving reduction, the inapproximability result for SETCOVER [9] carries over:

Theorem 2. *Approximating* CROSSINGSCHEDULE *within* $\mathcal{O}(\log n)$ *is NP-hard.*

3 Method

After exploring the combinatorial structure of the problem, we present an algorithm in which iteratively an MIP is solved. Details on the MIP formulation are given in Section 3.1.

Since we are considering homozygous ideotypes, we can assume without loss of generality that C^* has only 1-alleles and derive a lower bound based on the minimum set cover as follows. The universe corresponds to the loci, i.e. $U = \{1, \ldots, m\}$, and the subsets $\mathcal{S} = \{S_1, \ldots, S_n\}$ correspond to $\mathcal{P} = \{C^1, \ldots, C^n\}$. We define $p \in S_i$ if either $c^i_{1,p} = 1$ or $c^i_{2,p} = 1$ where $1 \leq i \leq n$ and $1 \leq p \leq m$. The following lemma now follows.

Lemma 2. *The cardinality of a minimum set cover is a lower bound on the number of crossings of any feasible crossing schedule*

Computing the minimum set cover is NP-hard. However, since in our experiments the number of loci and parents are relatively small, we are able to obtain the lower bound by solving a corresponding ILP [13] in a fraction of a second.

A lower bound on the population size can be obtained when considering the set $\mathcal{L}$ of all pairs of consecutive loci for which there are no genotypes in $\mathcal{P}$ containing 1-alleles at the respective loci on the same chromosome:

Lemma 3. *The following is a lower bound on the total population size.*

$$LB_{\text{pop}} = \sum_{(p,p+1) \in \mathcal{L}} N(r_{p,p+1}, \gamma) \tag{4}$$

Using (3) one can show that there is an optimal crossing schedules where homozygous genotypes are obtained via selfings.

Lemma 4. *There is an optimal schedule in which the (inner) homozygous genotypes are obtained via selfings.*

Finally, parental genotypes that contain a 1-allele at a locus at which all other parental genotypes contain all 0 have to be used by any feasible schedule. To reduce the search space explored by the MIP solver we fix these *compulsory* parental genotypes to be contained in any solution.

We present a MIP formulation for the problem variant where the number of crossings and the number of generations is fixed to F, respectively G. The reason for this is to be able to introduce cuts that ensure monotonically better solutions. In order to solve a problem instance, we iteratively consider combinations of (F, G) starting from $F = LB_{\text{crs}}$ and $G = 1 + \lceil \log_2 F \rceil$. In addition we enforce that the objective value of any feasible solution must be better than the currently best one. We do this by computing an upper bound UB_{pop} on the total population size, based on the best objective value found so far and the current values of (F, G) (see Algorithm 1, line 4). If at some point, say (F', G'), $LB_{\text{pop}} \geq UB_{\text{pop}}$ then we know that none of the combinations of $F'' \geq F'$, $G'' \geq G'$ will lead to a better solution. Therefore if $G = 1 + \lceil \log_2 F \rceil$ and $LB_{\text{pop}} \geq UB_{\text{pop}}$, we have found the optimal solution (see Algorithm 1, line 7). To guarantee termination for the case where $\lambda_{\text{crs}} = \lambda_{\text{gen}} = 0$, we stop incrementing F as soon as it reaches a pre-specified parameter UB_{crs}. Similarly, UB_{gen} is a pre-specified parameter bounding G. In Algorithm 1 the pseudo code is given.

Algorithm 1. OptCrossingSchedule(UB_{crs}, UB_{gen})

```
  Input: $UB_{\text{crs}}$ and $UB_{\text{gen}}$ are the maximum number of crossings and
         generations considered.
1 OPT $\leftarrow \infty$
2 for $F \leftarrow LB_{\text{crs}}$ to $UB_{\text{crs}}$ do
3     for $G \leftarrow 1 + \lceil \log_2 F \rceil$ to $\min(F, UB_{\text{gen}})$ do
4         $UB_{\text{pop}} \leftarrow \frac{1}{\lambda_{\text{pop}}}(\text{OPT} - F \cdot \lambda_{crs} - G \cdot \lambda_{gen})$
5         if $LB_{\text{pop}} < UB_{\text{pop}}$ then OPT $\leftarrow \min(\text{OPT}, \text{MIP}(F, G, UB_{\text{pop}}))$
6         else $UB_{\text{gen}} \leftarrow G - 1$
7     if $UB_{\text{gen}} \leq 1 + \lceil \log_2 F \rceil$ then return OPT
8 return OPT
```

3.1 MIP Formulation

Given an instance to CrossingSchedule with initial parental genotypes $\mathcal{P} = \{C^1, \ldots, C^n\}$, a feasible solution with G generations and F crossings can be characterized by the following five conditions: (i) The topology of the schedule is represented by a DAG with n source nodes $s_1, \ldots, s_n$, one target node t, and $F - 1$ additional nodes, where every non-source node has in-degree two. Parallel arcs are allowed and represent selfings. (ii) The longest path from a source node to the target node has length G. (iii) The alleles of each non-source node are derived from either the upper or lower chromosome of the node's respective predecessors. (iv) The genotype of a source node s_i is C^i, the genotype of t is C^*. (v) The probability of obtaining the genotype of an inner node v is at least $1-(1-\gamma)^{\frac{1}{N_{\max}}}$ such that its corresponding population size is at most $N_{\max}$. In the following we show how these conditions can be formulated as linear constraints. Throughout our formulation, we let $L := F + n$ be the total number of nodes. Dummies $1 \leq i, j \leq L$ correspond to genotypes, loci are indexed by $1 \leq p, q \leq m$ and chromosomes are referred to by $1 \leq k, l \leq 2L$. In the remainder of the paper we will omit the linearization of products of binary variables. Unless otherwise stated, we applied a standard transformation [1]. Similarly, we omit the details of the implementation of absolute differences of binary variables.

Feasibility constraints. The first set of constraints encodes the structure of the underlying DAG $D = (V, A)$. We assume a numbering of the vertices according to their topological order. In particular, arcs always go from vertices $j < i$ to a vertex i, $i, j \in V$. Based on the node numbering, the lower and upper chromosomes of a node $i \in V$ are respectively $2i - 1$ and $2i$. For convenience we introduce a mapping function $\delta(k)$ that returns the node a chromosome k corresponds to. Then binary variables $x_{k,i} \in \{0, 1\}$, $2n < k \leq 2L$, $i < \delta(k)$, denote whether chromosome k originates from genotype i, that is, they indicate an arc $(i, \delta(k))$. Since a chromosome originates from exactly one genotype, we have

$$\sum_{j=1}^{\delta(k)-1} x_{k,j} = 1 \qquad\qquad 2n < k \leq 2L \tag{5}$$

We capture the second condition by fixing a path of length G using the x variables and by restricting the depth of all remaining nodes, represented by additional integer variables, to be at most $G-1$. To model the third condition, we introduce binary variables $a_{k,p}$, $1 \leq k \leq 2L$, $1 \leq p \leq m$, which indicate the allele at locus p of chromosome k. Note that for chromosomes k corresponding to initial parental genotypes, $a_{k,p}$, $1 \leq p \leq m$, is a constant rather than a variable. In addition to knowing from which genotype a chromosome originates, we also need to know from which of the two chromosomes of that parental genotype an allele comes. Therefore we define binary variable $y_{k,p}$, $2n < k \leq 2L$, $1 \leq p \leq m$, to be 1 if the allele at locus p of chromosome k comes from the lower chromosome of its originating genotype; conversely $y_{k,p}$ is 0 if the allele originates from the upper chromosome. Now we can relate alleles to originating chromosomes. We do this by introducing binary variables $g_{k,p,l}$, for $2n < k \leq 2L$, $1 \leq p \leq m$, and $1 \leq l < 2\delta(k) - 1$. We define $g_{k,p,l} = 1$ if and only if the allele at locus p of chromosome k originates from chromosome l and has value 1. This is established through constraints

$$g_{k,p,2i} - a_{2i,p} \cdot x_{k,i} \cdot (1 - y_{k,p}) = 0 \qquad 2n < k \leq 2L, 1 \leq p \leq m, i < \delta(k) \tag{6}$$

$$g_{k,p,2i-1} - a_{2i-1,p} \cdot x_{k,i} \cdot y_{k,p} = 0 \qquad 2n < k \leq 2L, 1 \leq p \leq m, i < \delta(k) \tag{7}$$

Finally, an allele is 1 if and only if it originates from exactly one 1-allele:

$$a_{k,p} - \sum_{i=1}^{\delta(k)-1} (g_{k,p,2i-1} + g_{k,p,2i}) = 0 \qquad 2n < k \leq 2L, 1 \leq p \leq m \tag{8}$$

The fourth property can be ensured by simply forcing the variables representing the alleles of the parental genotypes and the alleles of the desired ideotype to the actual value of the respective allele. Thus for the parental genotypes we have $a_{2i-1,p} = c^i_{1,p}$ and $a_{2i,p} = c^i_{2,p}$ for $1 \leq i \leq n, 1 \leq p \leq m$ and for the ideotype $a_{2L-1,p} = a_{2L,p} = c^*_{1,p}$ for $1 \leq p \leq m$. The fifth property is enforced implicitly by the objective function.

Objective function. The probability of a given genotype i giving rise to a specific chromosome k determines the required population size (see (1)). This probability in turn depends on the exact set of crossovers necessary to generate chromosome k and on the sequence s of heterozygous loci (see (2)). Binary variable $\tilde{a}_{i,p} = 1$ if and only if locus p of genotype i is heterozygous: $\tilde{a}_{i,p} = |a_{2i-1,p} - a_{2i,p}|$ for $1 \leq i \leq L, 1 \leq p \leq m$. Now a genotype i is heterozygous, indicated by $h_i = 1$, if at least one of its loci is heterozygous: $h_i \geq \tilde{a}_{i,p}$ for $1 \leq i \leq L, 1 \leq p \leq m$. It is ensured that $h_i = 0$ whenever $\tilde{a}_{i,p} = 0$, $\forall 1 \leq p \leq m$, as $h_i = 1$ would increase the required population size. The distinction between the two different cases in (2) is based on crossover events between two successive heterozygous loci, i.e. $\nu(i)$ and $\nu(i+1)$. We capture the sequence s of heterozygous loci used in (2) by binary variables $b_{i,p,q}$, which indicate a maximal block of homozygous loci between heterozygous loci p and q in genotype i:

$$b_{i,p,q} = \tilde{a}_{i,p} \cdot \tilde{a}_{i,q} \cdot \prod_{r=p+1}^{q-1} (1 - \tilde{a}_{i,r}) \qquad 1 \leq i \leq L, 1 \leq p < q \leq m \tag{9}$$

To formulate the probability given in (2), let ξ_k^j denote the event of obtaining a chromosome k from a genotype j. Using variables h, b, and z, we can express $\Pr[\,\xi_k^j\,]$ such that in the heterozygous case every maximal homozygous block contributes $r_{p,q}$ if it contains at least one crossover, and $(1 - r_{p,q})$ otherwise. Finally, if j_1 and j_2 are the two parental genotypes of chromosomes k_1 and k_2 forming genotype i, we compute in variable $\bar{z}_i$ the log probability of event $\xi_{k_1}^{j_1} \cap \xi_{k_2}^{j_2}$ as $\ln(\Pr[\,\xi_{k_1}^{j_1}\,]) + \ln(\Pr[\,\xi_{k_2}^{j_2}\,])$. For that we have to sum over all possible $j < i$ to identify j_1 and j_2:

$$\begin{aligned} \bar{z}_i = \sum_{j<i} \sum_{l \in \{1,2\}} x_{k_l,j} \Bigg(h_j \ln(\frac{1}{2}) + \sum_{p=1}^{m-1} \sum_{q=p+1}^{m} b_{j,p,q} \ln(1 - r_{p,q}) \\ + \sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{r=p+1}^{q} b_{j,p,q} \cdot \ln(\frac{r_{p,q}}{1 - r_{p,q}}) \cdot |y_{k,r} - y_{k,r-1}| \Bigg) \end{aligned} \tag{10}$$

Notice that we neglect the possibility that the two chromosomes k_1 and k_2 may swap their originating genotypes as accounted for in the second case of equation (3) and we therefore might overestimate the population size. We will discuss this simplification in Section 5. Finally, we develop an approximation of the nonlinear function $N(\rho, \gamma)$ defining the required population size so that LP techniques can be utilized. More precisely, we reduce $N(\rho, \gamma)$ to a separable form [12] that depends only on a single decision variable and approximate it according to the *λ-method* [12] by a piecewise-linear curve specified by the points $(a_j, N(\mathrm{e}^{a_j}, \gamma))$ for $j = 1, \ldots, \ell + 1$. We replace the populations size $N(\mathrm{e}^{\bar{z}_i}, \gamma)$ for each crossing i in the objective function by a convex combination of the respective breakpoint scores to derive $\lambda_{\text{pop}} \cdot \left(\sum_{i=n+1}^{L} \sum_{j=1}^{\ell+1} \lambda_j^i \cdot N(\mathrm{e}^{a_j}, \gamma) \right) + \lambda_{\text{gen}} \cdot G + \lambda_{\text{crs}} \cdot F$.

Additional cuts. We consider three additional cuts. The first one is due to Lemma 4. The following constraints enforce that a homozygous genotype results via selfing: $|x_{2i-1,j} - x_{2i,j}| \leq h_j$ for $n < i \leq L, 1 \leq j < i$. In addition, the lower and upper bound on the population size correspond to $LB_{\text{pop}} \leq \sum_{i=n+1}^{L} \sum_{j=1}^{\ell+1} \lambda_j^i \cdot N(\mathrm{e}^{a_j}, \gamma) \leq UB_{\text{pop}}$ for $n < i \leq L$. For the sake of simplicity we omit the additional constraints required to enforce compulsory parental genotypes to be contained in the solution. To come back to condition five of our characterization of feasible solutions in the beginning of this section, we simply set $a_1 = \log(1 - (1 - \gamma)^{\frac{1}{N_{\max}}})$. Then any $\bar{z}_i < a_1$ implying a population size larger than $N_{\max}$ cannot be expressed as a convex combination of break points a_j, $j = 1, \ldots, \ell + 1$, and hence any feasible solution must satisfy the bound on the population size. In total, our MIP formulation comprises $\mathcal{O}(L(Lm^2 + \ell))$ many variables and $\mathcal{O}(L^2m)$ constraints.

4 Experimental Results

We have implemented OptCrossingSchedule in C++ using CPLEX 12.2[1] (default settings) with Concert Technology. We ran the experiments on a compute cluster with 2.26 GHz processors with 24 GB of RAM, running 64 bit Linux. We applied a time limit of 10 hours. Computations exceeding this limit were aborted. As mentioned earlier, there exist no previous methods for the general problem formulation we are considering. However, our problem formulation subsumes the one given by Servin et al., therefore we consider the same instances as well. In addition, we study a real-world instance. We conclude by evaluating our method on automatically generated instances. Throughout this section, the term 'provably optimal solution' indicates that the objective value of any feasible solution with respect to the piecewise-linear approximation and the simplification of (3) is at most the objective value of the obtained solution.

Instances by Servin et al. As opposed to our setting, in [10] a crossing schedule is required to be a tree. In addition, the number of initial parental genotypes $\mathcal{P} = \{C^0, C^1, \ldots, C^m\}$ is one more than the number of loci m. Parental genotypes are assumed to be homozygous. More specifically, C^0 consists of only 0-alleles, whereas for a genotype C^i, $1 \leq i \leq m$, the only 1-alleles are present at locus i. The ideotype is comprised entirely of 1-alleles and only the population size is considered, i.e. $\lambda_{\text{pop}} = 1, \lambda_{\text{gen}} = \lambda_{\text{crs}} = 0$. The desired probability of success is $\gamma = 0.999$ and the genetic distance between pairs of consecutive loci is 20 centimorgans (cM). By including constraints forcing a crossing schedule to be a tree (i.e. the out-degree of a node is forced to be 1), we were obtained the same optimal results (see Table 1). Servin et al. realize that better crossing schedules can be obtained when dropping the tree restriction. Rather than considering general DAGs, the authors consider a heuristic (PWC2) that transforms every enumerated tree into a DAG with smaller total population size. As opposed to the tree case, our method does not guarantee the solutions found in the DAG case to be optimal. This is because the objective function does neither include the number of crossings nor the number of generations. In addition, we put a

[1] http://www.cplex.com

Table 1. Results for instances by Servin et al. First column are the results on the tree cases (as obtained by Servin et al's method and our MIP), second column corresponds to PWC2 heuristic and the last column to our MIP for DAGs.

#loci	tree			PWC2			MIP		
	pop	crs	gen	pop	crs	gen	pop	crs	gen
4	374	5	5	359	7	5	350	5	5
5	551	6	6	516	8	6	482	9	8
6	770	7	7	691	9	6	624	9	7
7	1046	8	8	890	13	7	901	10	9
8	1394	9	9	1147	15	7	1329	10	10

time limit of 10 hours in place. In Table 1 we can see that we obtain better solutions w.r.t. the population size for the instances up to six loci. Due to the time limit, the best *feasible* solutions found for the instances with 7 and 8 loci are worse than the ones computed by Servin et al. Since PWC2 solutions are also feasible to our general model, a higher time limit would result in solutions that are at least as good as Servin's. We expect our approach to be less competitive with PWC2 on larger instances of this specific class. This comes at no surprise since PWC2 is specifically tailored toward these restricted instances.

Real-world instance. We consider a real-world case that deals with a disease in pepper called powdery mildew. This disease is caused by the fungus *Leveillula Taurica*. In severe cases of the disease the infected pepper plant may lose a significant amount of its leaves, which in turn results in crop loss. The fungus is resistant to fungicides, so host-plant resistance is desired. There is a wild-type pepper line that is resistant to the fungus. For this wild-type, three dominant quantitative trait loci (QTLs), numbered 1,2 and 3, that explain the resistance have been identified [11]. In addition to resistance, we also look at pungency, which is a dominant monogenic trait whose locus we assign number 4. The pungency gene is closely linked with one of the resistance QTLs, say the one of locus 3, with a genetic distance of 0.01 cM, i.e. $r_{3,4} = 0.01$ [5]. The resistant line is pungent. On the other hand, the elite line used for production is sweet but susceptible to the disease. Both lines are pure lines, i.e. they are homozygous at all loci. The goal now is to come up with a crossing schedule that results in a homozygous individual that is both resistant and sweet. We do this by using 1-alleles to indicate desired alleles. Therefore the parent set is $\mathcal{P} = \left\{ \begin{pmatrix} 1&1&1&0 \\ 1&1&1&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&1 \\ 0&0&0&1 \end{pmatrix} \right\}$, and the ideotype is $C^* = \begin{pmatrix} 1&1&1&1 \\ 1&1&1&1 \end{pmatrix}$. Unlinked loci by definition have a crossover probability of 1/2. So except for $r_{3,4}$, $r_{p,q} = 1/2$ for all $1 \leq p < q \leq 4$. We set $N_{\max} = 5000$ and $\gamma = 0.95$. Setting $\lambda_{\text{pop}} = 1/201, \lambda_{\text{gen}} = \lambda_{\text{crs}} = 100/201$ is a good trade off between the three criteria. In a practical setting, the λ-s are to be chosen such that they reflect the actual costs. Since there is a cost associated with the number of crossings and the number of generations, we are able to obtain a provably optimal solution in 1.5 seconds which is depicted (right) in Figure 1. It is important to note that this problem instance cannot be expressed in the restricted framework of Servin et al.[10]: treating the resistance loci as a

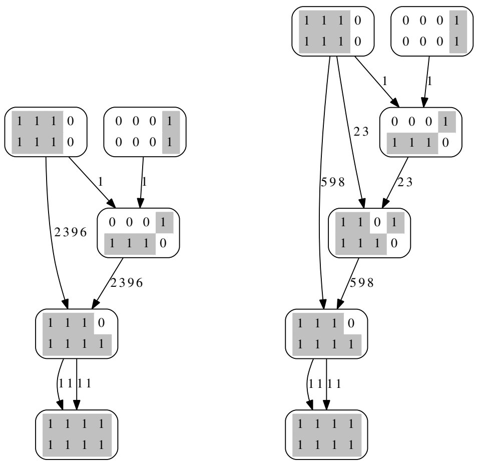
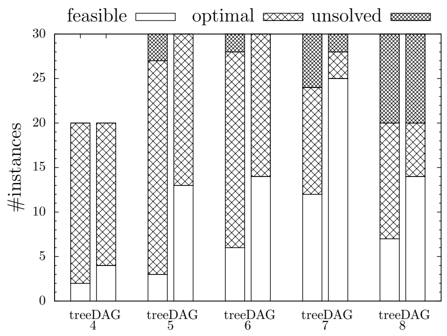


Fig. 1. Crossing schedules for the pepper instance. Inner nodes are obtained via crossings requiring a population size shown on the arcs, in both schedules the final crossing is a selfing. Chromosomes of an inner node are obtained via crossovers in their parents. *Left:* $F = 3$, $G = 3$, pop $= 2408$ and obj $= 14.69$. *Right:* provably optimal, $F = 4$, $G = 4$, pop $= 633$ and obj $= 7.13$.

single locus does not result in the best crossing schedule (see left of Figure 1), as the second genotype is obtained via a crossover between the second and third locus. To the best of our knowledge, such a real-world instance is solved for the first time to provable optimality within a precise mathematical model.

Generated instances. We generate random instances on which we evaluate the performance of our method. The instances either have 5 or 10 parents and concern 4-8 loci. The number of correct alleles per parental genotype affects the difficulty of the instances, we vary this number depending on the number of loci.



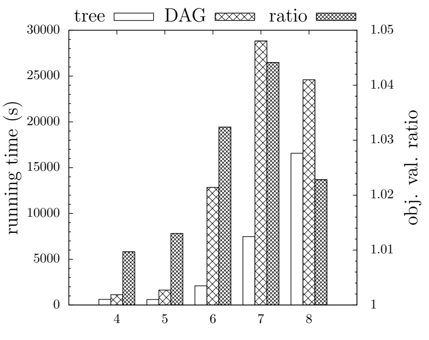


Fig. 2. Results for generated instances. *Left:* optimality of solutions. *Right:* running times; instances exceeding the time limit were not considered, objective value ratio (right y-axis).

In total 140 instances are generated, among which 20 concern instances of 4 loci; the classes of 5-8 loci are comprised by 30 instances each. We run both the DAG and the tree version of the MIP on all instances. For the DAG case, we were able to obtain solutions to 128 instances compared to 119 instances (see Figure 2) for the tree version. Among the unsolved instances for the tree case, there are also instances that are infeasible due to the value of $N_{\max}$ which requires re-use of genotypes. The number of instances that were solved to provable optimality in the DAG case is 58; for the tree case this number is 89. DAGs provide a gain in solution quality of up to 5% on average compared to the tree. Note that none of the instances is of the nature that is captured by Servin's model. Not surprisingly, trees are easier to solve.

5 Conclusion

For the first time we have described a mathematical model capturing the problem of marker-assisted gene pyramiding to its full extent. We show that our approach is capable of solving a real-world instance and generated instances, often to provable optimality. As mentioned earlier, our method is not exact due to (i) the piecewise-linear approximation of the population size function and (ii) a simplification in (10) of neglecting the possibility that the two chromosomes may swap their originating genotypes. However, in our experiments we have not observed any crossing where this could have happened. The NP-hardness proof involves only the number of crossings; as for the number of generations, the same reduction can be applied. The hardness with respect to the population size remains open. Possible extensions to our problem definition include considering heterozygous ideotypes. This requires an extension to tertiary alleles. Another extension would be to consider so called 'don't care' alleles, which are alleles that are not preserved due to crossover events, and as such do not need to be considered in the probability function.

Acknowledgments. We would like to thank Bertrand Servin for kindly providing us the source code of his method. In addition we are very grateful for the constructive comments of the anonymous referees.

References

1. Bradley, S.P., Hax, A.C., Magnanti, T.L.: Applied Mathematical Programming. Addison-Wesley, Reading (1977)
2. Brown, J., Caligari, P.: Introduction to Plant Breeding. Wiley-Blackwell (2008)
3. Collard, B.C.Y., Mackill, D.J.: Marker-assisted selection: an approach for precision plant breeding in the twenty-first century. Phil. Trans. R. Soc. B 363(1491), 557–572 (2008)
4. Dekkers, J.C.M., Hospital, F.: The use of molecular genetics in the improvement of agricultural populations. Nature Reviews Genetics 3, 22–32 (2002)
5. Haldane, J.B.S.: The combination of linkage values and the calculation of distances between the loci of linked factors. Journal of Genetics 8, 299–309 (1919)

6. Ishii, T., Yonezawa, K.: Optimization of the marker-based procedures for pyramiding genes from multiple donor lines: I. Schedule of crossing between the donor lines. Crop Science 47, 537–546 (2007)
7. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
8. Moose, S.P., Mumm, R.H.: Molecular plant breeding as the foundation for 21st century crop improvement. Plant Physiology 147, 969–977 (2008)
9. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In: Proc. 29th ACM Symp. on Theory of Computing, pp. 475–484 (1997)
10. Servin, B., Martin, O.C., Mézard, M., Hospital, F.: Toward a theory of marker-assisted gene pyramiding. Genetics 168(1), 513–523 (2004)
11. Shifriss, C., Pilowsky, M., Zacks, J.M.: Resistance to Leveillula Taurica mildew (=Oidiopsis taurica) in Capsicum annuum. Phytoparasitica 20(4), 279–283 (1992)
12. Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation and Application. Krieger Pub. Co. (1986)
13. Wolsey, L.A.: Integer programming. Wiley-Interscience series in discrete mathematics and optimization. Wiley, Chichester (1998)
14. Ye, G., Smith, K.F.: Marker-assisted gene pyramiding for inbred line development: Basic principles and practical guidelines. International Journal of Plant Breeding 2(1), 1–10 (2008)

Localized Genome Assembly from Reads to Scaffolds: Practical Traversal of the Paired String Graph

Rayan Chikhi and Dominique Lavenier

Computer Science department, ENS Cachan/IRISA, 35042 Rennes, France

Abstract. Next-generation de novo short reads assemblers typically use the following strategy: (1) assemble unpaired reads using heuristics leading to contigs; (2) order contigs from paired reads information to produce scaffolds. We propose to unify these two steps by introducing localized assembly: direct construction of scaffolds from reads. To this end, the paired string graph structure is introduced, along with a formal framework for building scaffolds as paths of reads. This framework leads to the design of a novel greedy algorithm for memory-efficient, parallel assembly of paired reads. A prototype implementation of the algorithm has been developed and applied to the assembly of simulated and experimental short reads. Our experiments show that our methods yields longer scaffolds than recent assemblers, and is capable of assembling diploid genomes significantly better than other greedy methods.

1 Introduction

De novo assembly of short reads consists in reconstructing a genome sequence given a set of short subsequences (*reads*) obtained from DNA sequencing. In practice, the original sequence cannot be retrieved unambiguously because a genome contains repetitions longer than the reads. Hence, one aims at finding a reasonable approximation of the sequence as a set of longer gap-less subsequences (*contigs*) or gapped subsequences (*scaffolds*). Over the last decade, three different genome assembly approaches have been adopted. Two of them are graph-based. The *string graph* method is based on a graph containing all the overlaps between reads [22]. The *de Bruijn graph* approach is based on the graph of all the k-length substrings of reads [24,21,13]. The third approach performs greedy extension of contigs using an ad-hoc structures [3,23]. Even producing an approximation of the genome is a computationally difficult task. For assembly of human-sized genomes using short reads ($<$ 100 bp), current state of the art implementations (using de Bruijn graphs) require hundreds of gigabases of memory and several CPU weeks of computation [13]. For more details concerning assembly approaches, refer to a recent survey [15].

Every next-generation sequencing technology is now able to massively produce *paired reads* separated by a known approximate distance (*insert size*). This information is highly valuable for re-sequencing projects as it enables better mapping coverage, especially with long inserts [6]. It also permits better resolution of repeats shorter than the insert size for de novo assembly. To illustrate the difference between single-end and paired assembly, an analogy with jigsaws can be made: consider n jigsaw pieces where each piece is linked to another piece by a string of finite length. The problem is to decide whether these pieces exactly fit into a $\sqrt{n} \times \sqrt{n}$ box with the additional constraint that every string must be tightened. In other words, pairing information indicates how far apart two pieces are in the solution. It can be shown that the paired jigsaw problem is also NP-complete. The reduction consists of splitting each classical jigsaw piece into 12 paired pieces.

Practically, pairs add more structure to the assembly problem, by indicating the relative position of reads on the genome. While most current assemblers use pairing information to improve assembly quality, they rely on single-end assembly beforehand. For instance, recent implementations of de Bruijn graph assembly methods use paired reads to simplify the graph once it is fully constructed and simplified from single-end reads [9,24,13]. The process of improving an existing assembly using paired reads is called *scaffolding.* The scaffolding problem is NP-complete and several heuristics have been proposed [18,11]. However, it may appear unsatisfactory to perform paired assembly using graph simplification or scaffolding, as such approach requires to solve unpaired assembly beforehand, which essentially ignores helpful pairing constraints.

Previous research has explored the benefits of using paired-end reads during contigs construction. The Arachne assembler searches for pairs of paired Sanger reads where both mates overlap to construct contigs [2]. The Shorty assembler uses pairing information to greedily construct contigs from paired reads anchored to long reads [10]. PE-Assembler extends contigs greedily and attempts to resolve ambiguous extensions using paired reads anchored nearby [1]. Medvedev et al. recently introduced the paired de Bruijn formalism, which incorporates pairing information in the de Bruijn graph [14]. Donmez et al. also recently proposed an approach to transform a string graph into a mate-pairs graph [7]. Each of these approaches aim to resolve repeats when constructing contigs. Our approach is essentially different as it uses paired reads for direct scaffold construction. One main advantage is that missing read overlaps (possibly due to sequencing artefacts, such as coverage gaps or localized errors) can be represented by gaps in scaffolds, whereas they would necessarily interrupt contigs. Note that all methods, including ours, do not implement mechanisms to resolve repetitions longer than the insert size.

In the next section, assembly of paired reads is formalized using the paired string graph representation. It is shown that scaffolds correspond to paths in the graph under ideal sequencing conditions. The definition of these paths is then refined to account for sequencing errors and biological variants. In section 3, a prototype implementation of path construction is applied to simulated and experimental sequencing data. A comparison is made with two other popular assemblers, using relevant assembly quality metrics.

2 Paired String Graph and Non-branching Paths

2.1 Paired Assembly Problem

The paired string graph is defined as an extension of the classical string graph [16] over a set of paired reads $R_1 \times R_2$. Two reads $(r, r') \in R_1 \cup R_2$ are said to *k-overlap* if a suffix of r matches a prefix of r' exactly over k characters. The *paired string graph* $PG^k(R_1 \times R_2)$ is defined as a directed graph by assigning a vertex to each read in $R_1 \cup R_2$. An edge $r \rightarrow r'$ is created between two reads if r k-overlaps r' (*overlap edge*). A special type of edge $r \dashrightarrow r'$ is created if (r, r') is a paired read (*paired edge*). Classical string graph transformations are applied: reads that are substrings of other reads, and transitively redundant overlap edges are discarded (paired edges are ignored during this step). No transitive reduction is performed for paired edges. For instance, consider the sequence $S = ab\mathbf{cd}ef\mathbf{cd}gh$ and perfect sequencing with insert of length 6 and paired reads of length 2. The paired string graph of these reads is drawn in Figure 1.

A *mixed path* in the paired string graph is a succession of vertices linked by either overlap edges or paired edges, e.g. $r_1 \rightarrow r_2 \dashrightarrow r_3 \rightarrow r_4$. *A path-string* is a string corresponding to the concatenation of nodes strings along a mixed path. The path-string is formed by the following rules: after an overlap edge, the string is appended with the concatenation of both nodes strings with their overlap repeated only once; after a paired edge, the string is appended with a gap corresponding to the paired insert size. In Figure 1, the path-string of $p = ab \rightarrow bc \dashrightarrow fc$ is $abc\Diamond^2 fc$, where $\Diamond$ is a single-character gap.

Similarly to the Assembly Problem (AP) [17], the *Paired Assembly Problem* can be defined as a constrained flavor of AP. The Paired Assembly Problem consists in finding a path that visits each node at least once (generalized Hamiltonian path) in $PG^k(R_1 \times R_2)$, and corresponds to a path-string s such that (1) the length of s is minimized and (2) for every pair (r, r') in $R_1 \times R_2$, the distance between r and r' in s matches the paired insert size. Note that a solution is necessarily a contig. Similarly to AP, this problem can also be shown to be

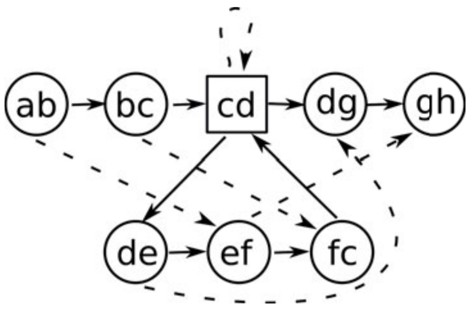


Fig. 1. Example of a paired string graph from paired reads (insert size of 6) covering the sequence $S = ab\mathbf{cd}ef\mathbf{cd}gh$. Dashes edges represent paired links and regular edges represent 1-overlaps between reads.

NP-hard. The following section focuses on constructing a collection of sub-paths (possibly scaffolds) that approximate a solution.

2.2 Non-branching Paths in the Ideal Case

Scaffolds can be directly constructed from the graph by following special types of mixed paths. To illustrate this, we first assume unrealistic sequencing conditions: error-free reads, perfect coverage and exact insert size (these will be relaxed in the next section). A mixed path in $PG^k(R_1 \times R_2)$ is a *non-branching path* (NBP) if each node, except the first and the last, has in-degree of 1 in the graph with respect to the corresponding in-edge type in the path, and out-degree of 1 corresponding to the out-edge type. In traditional assembly heuristics, a contig can be represented as a NBP where each edge is an overlap edge (*simple path*). For example, maximal-length contigs from the graph in Figure 1 are

$$\{ab \to bc \to cd, cd \to de \to ef \to fc \to cd, cd \to dg \to gh\}.$$

In contrast, a non-trivial non-branching path is

$$\{ab \dashrightarrow ef \dashrightarrow gh\},$$

where the path-string $(ab\Diamond^2 ef\Diamond^2 gh)$ is a scaffold which covers the whole string. Under ideal sequencing conditions, non-branching paths immediately correspond to valid scaffolds. One can also consider *in- (resp. out-) branching paths*, for which only out- (resp. in-) degree of nodes in the path with respect to the corresponding edge type is 1. By similar reasoning, it can be shown that such paths also spell valid scaffolds.

2.3 Practical Non-branching Paths

In actual sequencing, we distinguish two situations: undetected paired branching and additional overlap branching. Previously, paired branching was always detected because of perfect coverage and exact insert size. Now, it is no longer sufficient for a node to have an unique paired edge in order to unambiguously extend a scaffold. Weaker conditions can be formulated to detect the absence of paired branching, given imperfect coverage and variable insert size. First, assume that the insert size deviation is bounded by a constant i. Second, consider a simple path p of length $2i + 1$, and let n be the central node (p_{i+1}).

Property 1. A paired edge $n \dashrightarrow n'$ is considered to satisfy the non-branching condition if the sub-graph induced by the opposite mates of nodes in p is a simple path p' of central node $n' = p'_{\lfloor \frac{|p'|}{2} \rfloor}$.

In other words, it is possible to detect that p' is the only genomic region which appears at approximately an insert distance further than p. The original definition of non-branching paths can then be extended to include this condition in place of the paired degree condition.

Furthermore, sequencing errors and biological variants introduce additional branching in the graph. The branching structures are referred as *bubbles* (multiple paths that starts and ends at the same nodes) and *tips* (short interrupted paths) [24]. Graph-based assembly algorithms typically remove bubbles and tips after the whole graph is constructed. Here, the bubble detection technique presented in [24] is adapted to also detect tips. As these structures are short, one can set a maximal length $d > 0$ for paths within them. A general characterization of these structure can be made in terms of sub-graph traversal. Observe that both structures form sub-graphs which start at a single node, and paths which are not interrupted converge after a certain length.

Property 2. Given a variant sub-graph, the breadth-first tree constructed from the start node has a single node of depth d.

Non-branching paths are extended to permit traversal in short branching sub-graphs through overlap edges. Figure 2 illustrates both properties. In summary, we define practical non-branching paths (PNBP) as follows:

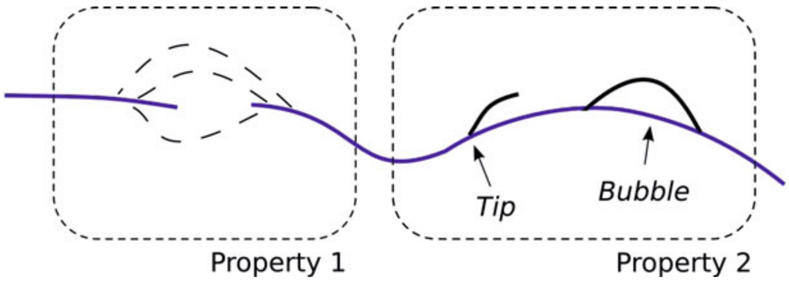


Fig. 2. Practical non-branching path traversal (blue line) of a paired string sub-graph. Thick lines represent paths of overlap edges. Dashed lines represent paired edges between reads. Property 1 is used to traverse a gap, as paired reads link together two simple paths. Property 2 is used to traversal small branching regions (a tip and a bubble).

– for path nodes $n \dashrightarrow m$ linked by a paired edge, both n and m are middle nodes of simple paths of length $2i + 1$ for which Property 1 is verified.
– for path nodes $n \rightarrow m$ linked by an overlap edge, either the overlap out-degree of n and the overlap in-degree of m are both 1, or $n \rightarrow m$ is part of a sub-graph which satisfies Property 2.

Note that setting $i = 0$ and $d = 1$ corresponds to the original definition of non-branching paths. Practical in-branching (resp. out-branching) paths are defined similarly, except that Property 1 only needs to be verified for n (resp. m).

2.4 Localized, Parallel Assembly

For large genomes, constructing the classical string graph is a memory-intensive operation. This issue also applies to paired string graphs, as they contain strictly more information. Two possible solutions are considered to reduce memory usage. A compressed FM-index [8] of the reads reduces the memory usage of the string graph [20]. This approach could be extended to include paired edges, computed dynamically from indexed paired reads to avoid memory overhead.

Another solution is to perform localized assembly, without initial construction of the whole paired string graph. The key property of practical non-branching paths is that only a small sub-graph needs to be explored for each path. We propose to take advantage of this property to perform assembly both locally and in parallel. Concretely, a greedy assembly algorithm can start from any read and construct a sub-graph of the paired string graph on the fly, by following a PNBP strategy. This is equivalent to splitting the complete paired string graph into disjoint sub-graphs, each sub-graph corresponds to exactly one scaffold. This approach induces a memory overhead due to the parallel construction of sub-graphs, however only a constant number of scaffolds is assembled in parallel at any given time. To construct each sub-graph, overlaps between paired reads and pairing information need to be accessed efficiently. Hence, it is required that a complete index of the reads resides in memory. However, such index occupies less memory than a string graph. Provided that each worker can access the full index, embarrassingly parallel construction of scaffolds can be achieved.

While it is technically a greedy algorithm, such approach overcomes the shortcomings of classical greedy algorithms. Except for Taipan [19], which does not support paired reads nor parallel assembly, most greedy assemblers do not construct a graph to solve local extension ambiguities [4,1,3]. As a consequence, greedy assemblers stop contigs extension at small biological variations or sequencing artifacts. Localized graph-based assembly using Property 2 overcomes this problem by explicitly performing traversal of such structures.

3 Results

We developed a prototype assembler called Monument based on local construction of practical in-branching paths. The prototype is implemented in the Python language with C++ extensions for critical parts. For memory efficiency, a kmer-based reads indexing structure is used. Specifically, the indexing procedure minimizes the number of reads referenced by each kmer, while still maintaining branching information. For practical in-branching paths, the maximal graph depth d for genomic variants is set such that any path has genomic length less than $10 + k$. The insert size deviation i is set to half the value of the insert size, which is a very conservative deviation with respect to actual paired-end data. To ensure fair comparison with other methods, we implemented a naive gap-closing procedure which fill scaffolds gaps with any overlap path satisfying insert size constraints.

Two short reads assemblers based on de Bruijn graphs are compared with this prototype. The Velvet assembler (version 1.1.03) uses graph simplification heuristics [24]. The Ray assembler (version 1.3.0) implements a greedy traversal strategy [3]. The assemblers were run with default parameters and $k = 23$. By setting a similar kmer size, all assemblers, including ours, virtually explore the same de Bruijn graph.

We first compared assemblers on experimental Illumina short paired reads from *E. coli* (SRA SRX000429). This dataset (Dataset 1) contains 10 million paired reads of length 36 bp and insert size 200 bp. We then investigated the ability of our method to assemble diploid genomes. To this end, we simulated 3 million paired reads of a diploid genome based on the *E. coli* sequence (Dataset 2). The wgsim paired reads simulator was used with default parameters [12], producing 75 bp reads (500 bp inserts) with simulated sequencing errors. Assembly results for the datasets are shown in Table 1. To understand why Ray has difficulties assembling the second dataset, we simulated a third dataset of reads, similar to Dataset 2 but without variants. This time, Ray obtains a scaffold N50 of 89.4 Kbp and largest scaffold of length 268.5 Kbp. This experiment confirms that mechanisms for biological variations traversal, such as PNBPs, are a key requirement for greedy assemblers.

Table 1. Quality of the assemblies of simulated and experimental paired-end reads from *E.coli* using Velvet, Ray and our prototype (Monument). The N50 metric measures the length of the smallest element of the set of largest scaffolds (resp. contigs) which cover at least 50% of the assembly. Coverage and accuracy of scaffolds are assessed using evaluation tools from Allpaths [5]. Specifically, scaffolds are divided into chunks of size less than 10kb. Considering the high coverage of the datasets, each chunk is considered to be valid if it aligns with more than 99% identity to the reference genome (alignment with undertermined nucleotides are considered valid).

Dataset	Software	Contig N50 (Kbp)	Scaffold N50 (Kbp)	Longest scaffold (Kbp)	Coverage (%)	Accuracy (%)
Experimental (1)	Monument	38.0	101.8	236.0	96.4	96.7
	Velvet	26.3	95.3	267.9	96.9	99.1
	Ray	69.5	87.3	174.4	97.4	98.4
Simulated with variants (2)	Monument	113.3	134.1	340.5	91.0	95.0
	Velvet	30.8	132.6	327.2	87.9	92.3
	Ray	10.2	10.2	41.2	89.2	100.0

We recorded execution time and memory usage during the assembly of the experimental dataset. The size of the paired reads index is 0.4 GB and peak memory usage during assembly is 0.6 GB. Velvet and Ray have peak memory usage of 2.4 GB and 3.2 GB respectively. However, Ray can distribute its indexing structure on a cluster. Using 6 threads, our implementation completed the assembly in 7 minutes, Velvet in 8 minutes and Ray in 16 minutes.

Our implementation can also assemble a scaffold around a specified genomic region, i.e. perform targeted assembly. This is of particular interest as new targeted assembly methods (TASR and Mapsembler, both unpublished) only produce contigs. Targeted assembly with our prototype is also very fast: one scaffold is assembled in a few seconds. However, contrary to targeted assemblers, the prototype requires the complete reads index to reside in memory.

4 Discussion

In summary, a new *de novo* assembly framework is formulated by introducing paired string graphs. The novelty of this framework resides in its ability to perform localized assembly of scaffolds. Prior to this work, scaffolds were constructed from an ordering of contigs, requiring a complete assembly of contigs to be known beforehand. We show that it is possible to assemble scaffolds locally around a genomic region by following non-branching paths greedily. This approach allows to design the first localized assembly algorithm which directly constructs scaffolds from reads.

Compared to other greedy approaches, our approach takes into account biological variants. Hence, it does not suffer from degraded contiguity with diploid genomes. Preliminary benchmark results on simulated and experimental datasets indicate that this method yields longer scaffolds than two leading short reads assemblers. We conjecture that scaffolders implemented in popular assemblers do not take full advantage of the whole contigs graph, as our greedy traversal obtains comparable results. Additionnally, practical benefits of this algorithm are twofold. It is embarrassingly parallelizable, as scaffolds can be constructed independently. It also does not require a large graph to be stored in memory, a small graph is constructed for each scaffold.

A natural future direction for this work is to compare the prototype with more existing tools, especially string graphs implementations, on larger datasets. Two other aspects should also be considered: (1) gap-closing in scaffolds is a key step for obtaining long contigs. Most complex repeats were not resolved by our simple path-finding procedure, hence a more elaborate algorithm is needed. (2) Incorporating mate-pairs with long inserts in genomic graphs is still an unaddressed challenge in the literature. These reads are produced with higher insert size variability and lower coverage than paired-end reads. Mate-pairs cannot be used in our current framework, because Property 1 almost never holds for such data. An immediate solution would be to perform re-scaffolding of scaffolds using mate-pairs links.

As short read sequencing is progressively shifting towards longer reads (over 100 bp), the landscape of assembly software has to adapt to high-coverage, longer reads. Specifically, de Bruijn graph implementations appear to be unable to assemble long reads with quality comparable to string graph implementations. In contrast, string graph-based methods are limited to assembly of low-volume datasets because of memory constraints. We believe that our methodology will lead to software able to assemble both short and long reads at any coverage without sacrificing running time or results quality.

Acknowledgements. The authors are grateful to J. Nicolas for helpful discussions. This work benefited from the ANR grant associated with the MAPPI project (2010-2014).

References

1. Ariyaratne, P.N., Sung, W.: PE-Assembler: de novo assembler using short paired-end reads. Bioinformatics (December 2010)
2. Batzoglou, S., Jaffe, D.B., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J.P., Lander, E.S.: ARACHNE: a whole-genome shotgun assembler. Genome Research 12(1), 177 (2002)
3. Boisvert, S., Laviolette, F., Corbeil, J.: Ray: Simultaneous assembly of reads from a mix of High-Throughput sequencing technologies. Journal of Computational Biology, 3389–3402 (2010)
4. Bryant, D.W., Wong, W.K., Mockler, T.C.: QSRA – a quality-value guided de novo short read assembler. BMC Bioinformatics 10(1), 69 (2009)
5. Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I.A., Belmonte, M.K., Lander, E.S., Nusbaum, C., Jaffe, D.B.: ALLPATHS: de novo assembly of whole-genome shotgun microreads. Genome Research 18(5), 810–820 (2008), http://genome.cshlp.org/content/18/5/810.abstract
6. Chikhi, R., Lavenier, D.: Paired-end read length lower bounds for genome re-sequencing. BMC Bioinformatics 10(suppl. 13), O2 (2009)
7. Donmez, N., Brudno, M.: Hapsembler: An assembler for highly polymorphic genomes. In: Bafna, V., Sahinalp, S.C. (eds.) RECOMB 2011. LNCS, vol. 6577, pp. 38–52. Springer, Heidelberg (2011)
8. Ferragina, P., Manzini, G.: Indexing compressed text. Journal of the ACM (JACM) 52(4), 552–581 (2005)
9. Gnerre, S., MacCallum, I., Przybylski, D., Ribeiro, F.J., Burton, J.N., Walker, B.J., Sharpe, T., Hall, G., Shea, T.P., Sykes, S., Berlin, A.M., Aird, D., Costello, M., Daza, R., Williams, L., Nicol, R., Gnirke, A., Nusbaum, C., Lander, E.S., Jaffe, D.B.: High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Proceedings of the National Academy of Sciences 108(4), 1513–1518 (2011), http://www.pnas.org/content/108/4/1513.abstract
10. Hossain, M., Azimi, N., Skiena, S.: Crystallizing short-read assemblies around seeds. BMC Bioinformatics 10(suppl. 1), S16 (2009), http://www.biomedcentral.com/1471-2105/10/S1/S16
11. Huson, D.H., Reinert, K., Myers, E.W.: The greedy path-merging algorithm for contig scaffolding. Journal of the ACM (JACM) 49(5), 603–615 (2002)
12. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R.: The sequence alignment/map format and SAMtools. Bioinformatics 25(16), 2078 (2009)
13. Li, R., Zhu, H., Ruan, J., Qian, W., Fang, X., Shi, Z., Li, Y., Li, S., Shan, G., Kristiansen, K., Li, S., Yang, H., Wang, J., Wang, J.: De novo assembly of human genomes with massively parallel short read sequencing. Genome Research 20(2), 265–272 (2010), http://genome.cshlp.org/content/20/2/265.abstract
14. Medvedev, P., Pham, S., Chaisson, M., Tesler, G., Pevzner, P.: Paired de bruijn graphs: A novel approach for incorporating mate pair information into genome assemblers. In: Bafna, V., Sahinalp, S.C. (eds.) RECOMB 2011. LNCS, vol. 6577, pp. 238–251. Springer, Heidelberg (2011)

15. Miller, J.R., Koren, S., Sutton, G.: Assembly algorithms for next-generation sequencing data. Genomics (2010)
16. Myers, E.W.: Toward simplifying and accurately formulating fragment assembly. Journal of Computational Biology 2(2), 275–290 (1995)
17. Nagarajan, N., Pop, M.: Parametric complexity of sequence assembly: Theory and applications to next generation sequencing. Journal of Computational Biology 16(7), 897–908 (2009)
18. Pop, M., Kosack, D.S., Salzberg, S.L.: Hierarchical scaffolding with bambus. Genome Research 14(1), 149–159 (2004), http://genome.cshlp.org/content/14/1/149.abstract
19. Schmidt, B., Sinha, R., Beresford-Smith, B., Puglisi, S.J.: A fast hybrid short read fragment assembly algorithm. Bioinformatics 25(17), 2279 (2009)
20. Simpson, J.T., Durbin, R.: Efficient construction of an assembly string graph using the FM-index. Bioinformatics 26(12), i367 (2010)
21. Simpson, J., Wong, K., Jackman, S., Schein, J., Jones, S., Birol, İ.: ABySS: A parallel assembler for short read sequence data. Genome Research 19(6), 1117 (2009)
22. Sutton, G., Miller, J.R., Delcher, A.L., Koren, S., Venter, E., Walenz, B.P., Brownley, A., Johnson, J., Li, K., Mobarry, C.: Aggressive assembly of pyrosequencing reads with mates. Bioinformatics 24(24), 2818–2824 (2008), http://bioinformatics.oxfordjournals.org/cgi/content/abstract/24/24/2818
23. Warren, R.L., Sutton, G.G., Jones, S.J.M., Holt, R.A.: Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23(4), 500–501 (2007), http://bioinformatics.oxfordjournals.org/cgi/content/abstract/23/4/500
24. Zerbino, D.R., Birney, E.: Velvet: Algorithms for de novo short read assembly using de bruijn graphs. Genome Research 18(5), 821–829 (2008), http://genome.cshlp.org/content/18/5/821.abstract

Stochastic Errors vs. Modeling Errors in Distance Based Phylogenetic Reconstructions (Extended Abstract)

Daniel Doerr[1], Ilan Gronau[2], Shlomo Moran[3], and Irad Yavneh[3]

[1] Center for Biotechnology, Bielefeld University, Bielefeld, Germany
ddoerr@cebitec.uni-bielefeld.de

[2] Dept. of Biological Statistics and Computational Biology, Cornell University, Ithaca, USA
ig67@cornell.edu

[3] Computer Science Dept., Technion - Israel Institute of Technology, Haifa, Israel
{moran,irad}@cs.technion.ac.il

Abstract. Distance based phylogenetic reconstruction methods use the *evolutionary distances* between species in order to reconstruct the tree spanning them. This paper continues the line of research which attempts to adjust to each given set of input sequences a distance function which maximizes the expected accuracy of the reconstructed tree. We demonstrate both analytically and experimentally that by deliberately assuming an oversimplified evolutionary model, it is possible to increase the accuracy of reconstruction.

1 Introduction

Distance based methods for reconstruction of phylogenetic trees from biological sequences usually follow the following four steps: (1) a substitution model of sequence evolution is assumed; (2) a *substitution rate* (SR) function Δ is selected (Δ typically corresponds to *additive* distances in the assumed model); (3) the $\binom{n}{2}$ interspecies distances defined by Δ are estimated from alignments of the n input sequences; (4) a tree that best fits the estimated distances is constructed over the n input species.

There are two main sources for inaccuracies in the above reconstruction process: (a) a wrong model chosen in (1) could imply that the function Δ selected in (2) is not additive for the true model; (b) *stochastic errors* associated with the estimation of distances from alignments of finite length in (3). In previous works [9,10] we have shown that most common DNA substitution models (except for Jukes-Cantor) have many different additive SR functions with different patterns of stochastic error. We demonstrated that selecting a function that is expected to be least noisy for the given input leads to significant improvement in the accuracy of the reconstructed tree. In this paper we extend this line of research to cases where the selected model is not the true model. Somewhat surprisingly, we show both analytically and via experiments on real and simulated data, that by deliberately assuming an oversimplified evolutionary model and using a non-additive but less noisy SR function, it is possible to increase the accuracy of reconstruction. This result appears to be related to the following well known phenomenon: the Maximum Parsimony reconstruction method, which is not statistically consistent in general, provides a higher reconstruction accuracy rate in certain cases compared to reconstruction methods that are statistically consistent (see, e.g., [20,18,6]).

In Section 2 we present the required background and the concept of *affine-additive* mappings. In Section 3 we present the concept of *deviation from additivity* which measures the deviation of a non-affine-additive mapping from the closest affine-additive one. Section 4 demonstrates the possible advantage of using an oversimplified model when reconstructing quartets by the four-points method. We then present a useful heuristic, based on Fisher's linear discriminant, for identifying scenarios in which such oversimplification is useful. In Section 5 we give a brief demonstration of our approach in reconstruction of phylogenies from real biological sequences.

Note: Due to space limitations, most of the formal analysis and some experimental results are omitted from this extended abstract and can be found in [4].

2 Background

A DNA substitution model $\mathcal{M}$ consists of a set of stochastic 4×4 *transition matrices* (describing possible substitution patterns) closed under matrix product (i.e., $\mathbf{P}, \mathbf{Q} \in \mathcal{M} \rightarrow \mathbf{P} \cdot \mathbf{Q} \in \mathcal{M}$). These matrices serve to describe the substitution process along evolutionary paths in a phylogenetic tree. A *model tree* in a substitution model $\mathcal{M}$ is an undirected tree $T = (V, E)$ in which each edge $e \in E$ is associated with a transition matrix $\mathbf{P}_e \in \mathcal{M}$. A model tree T implies an inter-leaf transition matrix $\mathbf{P}_{ij} \in \mathcal{M}$ for each pair of leaves $\{i, j\} \subset L(T)$. In most common substitution models, each substitution matrix $\mathbf{P}$ is given as an exponentiation $\mathbf{P} = e^{\mathbf{R}}$ of a *rate matrix* $\mathbf{R}$, where $\mathbf{R}$ is a 4×4 matrix whose off-diagonal elements are non-negative, and whose rows sum to 0. A *homogeneous* substitution model is a model defined by a fixed *unit rate matrix* $\mathbf{R}$. i.e.: $\mathcal{M}_{\mathbf{R}} = \{e^{t\mathbf{R}} \ : \ t \in \mathbb{R}^+\}$. Homogeneous model trees are thus defined by a set of edge-rates $\{t_e\}_{e \in E(T)}$ (t_e is the evolutionary time associated with e). A *substitution rate (SR) function* for a model $\mathcal{M}$ is a non-negative function $\Delta : \mathcal{M} \rightarrow \mathbb{R}^+$ that maps each transition matrix onto a numerical value of "substitution rate". An SR function Δ induces the following "dissimilarity mapping" over the leaves of a model tree T in $\mathcal{M}$: $D^T_\Delta(i, j) = \Delta(\mathbf{P}_{ij})$, for all $\{i, j\} \subset L(T)$. Of particular interest in phylogenetic reconstruction are *additive* SR functions.

Definition 1 (Additive SR function). *An SR function Δ is said to be* additive *for a substitution model $\mathcal{M}$ if for all $\mathbf{P}, \mathbf{Q} \in \mathcal{M}$, $\Delta(\mathbf{PQ}) = \Delta(\mathbf{P}) + \Delta(\mathbf{Q})$.*

Let Δ be an additive SR function. Then for each model tree T, the pairwise distances $\{D_\Delta(i, j) = \Delta(\mathbf{P}_{ij}) \ : \ i, j \in L(T)\}$ induce an *additive metric* on T, which associates a positive weight $w(e)$ for each edge $e \in T$. Moreover, the tree T and the edge-weights $\{w(e) : e \in T\}$ can be restored from the matrices $\{\mathbf{P}_{ij} : i, j \in T\}$. This fact is at the heart of the common approach for distances estimation, which starts by obtaining estimates $\{\hat{\mathbf{P}}_{ij}\}_{i,j\in L(T)}$ of the inter-leaf transition matrices $\mathbf{P}_{ij}$ (through maximum likelihood), and then estimates pairwise distances using some additive SR function Δ: $\hat{D}_\Delta(i,j) = \Delta(\hat{\mathbf{P}}_{ij})$. Statistical consistency of the reconstruction relies on the fact that as the sequence length grows, the estimated matrices $\hat{\mathbf{P}}_{ij}$ converge to the true matrices $\mathbf{P}_{ij}$, and the estimated distances $\hat{D}_\Delta(i,j)$ converge to the additive distances $D_\Delta(i,j)$. Accurate reconstruction of the topology of the tree is guaranteed when the distance estimation errors are smaller than half the minimal weight of an internal edge in the tree [1,5].

The above properties of additive SR functions are easily extended to affine transformation $\Delta_{aff} = a\Delta + b$, where Δ is additive, $a \in \mathbb{R}^+$ and $b \in \mathbb{R}$. This is due to a simple technical observation that such transformations "preserve" the internal edge-weights of the implied weighted tree [4]. Such affine transformations of additive SR functions are termed *affine-additive mappings*, and as we demonstrate are very useful in analyzing the accuracy and consistency of non-additive SR functions, which is the main theme of this paper.

2.1 SR Functions for the Kimura 2-Parameter Model

We use Kimura's two-parameter model (K2P) [13] as a simple case study throughout the paper. K2P is a DNA substitution model in which each rate matrix is defined by two parameters: α, β. The α parameter describes the rate of *transition*-type (ti) substitutions (`A` $\leftrightarrow$ `G`, `C` $\leftrightarrow$ `T`) and the β parameter describes the rate of *transversion*-type (tv) substitutions ($\{$`A,G`$\} \leftrightarrow \{$`C,T`$\}$).

$$\mathcal{M}_{\text{K2P}} = \left\{e^{\mathbf{R}_{\alpha,\beta}} \mid \alpha \geq \beta > 0\right\} \quad ; \quad \mathbf{R}_{\alpha,\beta} = \begin{pmatrix} - & \alpha & \beta & \beta \\ \alpha & - & \beta & \beta \\ \beta & \beta & - & \alpha \\ \beta & \beta & \alpha & - \end{pmatrix} \tag{1}$$

We assume a normalization scheme in which each rate matrix is factored as follows: $\mathbf{R}_{\alpha,\beta} = t\mathbf{R}_{\alpha',\beta'}$, where $t = \alpha + 2\beta$ and $\alpha' + 2\beta' = 1$. We refer to $\mathbf{R}_{\alpha',\beta'}$ and t as the *unit rate matrix* and *evolutionary time* (resp.) associated with $\mathbf{R}_{\alpha,\beta}$. The matrix exponentiation $e^{\mathbf{R}_{\alpha,\beta}}$ results in a stochastic transition matrix with p_α indicating the probability of a transition-type substitution and p_β indicating the probability of a transversion-type substitution. The transformations between (α, β) and (p_α, p_β) are given by:

$$\alpha = -\frac{1}{2}\ln(1 - 2p_\beta - 2p_\alpha) + \frac{1}{4}\ln(1 - 4p_\beta) \qquad \beta = -\frac{1}{4}\ln(1 - 4p_\beta) \ . \tag{2}$$

$$p_\alpha = \frac{1}{4}\left(1 + e^{-4\beta} - 2e^{-2\alpha-2\beta}\right) \qquad p_\beta = \frac{1}{4}\left(1 - e^{-4\beta}\right) \ . \tag{3}$$

A *homogeneous sub-model* of $\mathcal{M}_{\text{K2P}}$ consists of all rate matrices proportional to some unit K2P rate matrix, hence all rate matrices in such a model share the same ti-tv ratio $R = \frac{\alpha}{2\beta} \geq \frac{1}{2}$. The Jukes-Cantor (JC) model [12] is a special homogeneous sub-model of $\mathcal{M}_{\text{K2P}}$ in which $R = \frac{1}{2}$. In this paper we study two SR functions in $\mathcal{M}_{\text{K2P}}$:

$$\varDelta_{\text{K2P}}(p_\alpha, p_\beta) = -\frac{1}{2}\ln(1 - 2p_\beta - 2p_\alpha) - \frac{1}{4}\ln(1 - 4p_\beta) \;=\; \alpha + 2\beta \;. \tag{4}$$

$$\varDelta_{\text{JC}}(p_\alpha, p_\beta) \;= -\frac{3}{4}\ln\left(1 - \frac{4}{3}(p_\alpha + 2p_\beta)\right) \;=\; \frac{3}{4}\ln\left(\frac{1}{3}(e^{-4\beta} + 2e^{-2\alpha-2\beta})\right) \tag{5}$$

$\varDelta_{\text{K2P}}$ is the common (additive) SR function used in the general context of $\mathcal{M}_{\text{K2P}}$, as suggested in [13]. $\varDelta_{\text{JC}}$ coincides with $\varDelta_{\text{K2P}}$ when the ti-tv ratio is $R = \frac{1}{2}$, but it is non-affine-additive in all other homogeneous sub-models of $\mathcal{M}_{\text{K2P}}$.

3 Deviation from Additivity in Homogeneous Substitution Models

In this section we briefly review a general framework for analyzing non-affine-additive SR functions in a homogeneous substitution model. In a homogeneous model $\mathcal{M}_{\mathbf{R}}$, defined by a *unique* unit rate matrix $\mathbf{R}$, each rate matrix has the form $t\mathbf{R}$, and can thus be directly associated with the evolutionary time parameter t. It is useful to view an SR function for $\mathcal{M}_{\mathbf{R}}$ as a one-dimensional function $\varDelta$ which maps the evolutionary time t (rather than the matrix $e^{t\mathbf{R}}$) to a dissimilarity measure $\varDelta(t)$. Viewed this way, an SR function $\varDelta$ is affine-additive for the homogeneous model if and only if $\varDelta(t) = at + b$ for some $a \in \mathbb{R}^+, b \in \mathbb{R}$.

In order to determine whether a non-affine-additive SR function implies consistent reconstruction of a given model tree, we introduce the concept of *deviation from additivity.* The *deviation* of an SR function $\varDelta$ from a given affine-additive function $at+b$ in an interval $[t_0, t_1]$ is defined as $\frac{1}{a}\max\{|\varDelta(t) - at - b| \,:\, t \in [t_0, t_1]\}$ (the factor $\frac{1}{a}$ normalizes the deviation to units of evolutionary time). The *deviation of $\varDelta$ from additivity within* $[t_0, t_1]$ is the minimum deviation of $\varDelta$ from any affine-additive function in that interval. Relying on the results of Atteson [1], we prove in [4] that an SR function $\varDelta$ can be used to consistently reconstruct the topology of a model tree when its deviation from additivity in the appropriate interval (defined by the minimum and maximum inter-leaf evolutionary times of the tree) is less than $\frac{1}{2}t_{\min}$, where $t_{\min}$ is the evolutionary time associated with a shortest edge in the model tree. We also show that for many common types of functions, for instance convex or concave functions, the linear interpolation of $\varDelta$ can be used to minimize its deviation.

In our K2P case study, $\varDelta_{\text{JC}}$ is a concave SR function in any homogeneous sub-model of $\mathcal{M}_{\text{K2P}}$. Figure 1a depicts $\varDelta_{\text{JC}}$ and its linear interpolation $\varDelta_{int}$ in the interval $[0.4, 1.4]$ when the ti-tv ratio is $R = 10$. X in that figure denotes the value $\max\{|\varDelta_{\text{JC}}(t) - \varDelta_{int}(t)| : t \in [0.4, 1.4]\}$. Figure 1b shows $\varDelta_{\text{JC}}$ in the same setting with the SR function $\varDelta_{\text{int}} + \frac{1}{2}X$, which is the affine-additive function minimizing the deviation of $\varDelta_{\text{JC}}$ in that interval. In that figure we also plot the *stochastic* error margins, corresponding to the first-order approximation of the standard deviation of these two functions. This approximation is obtained in [9] by applying the delta method, and is inversely proportional to the sequence length (the plots in Fig. 1b assume a sequence length of 500 bp). Note how the margins of Δ_{JC} are actually more tightly concentrated around its affine-additive approximation $\Delta_{int}+\frac{1}{2}X$ than the margins of that affine-additive approximation. This implies that in this setting, distances obtained by using the non-affine-additive Δ_{JC} are actually more likely to be near-additive than distances obtained by using the additive SR function Δ_{K2P}.

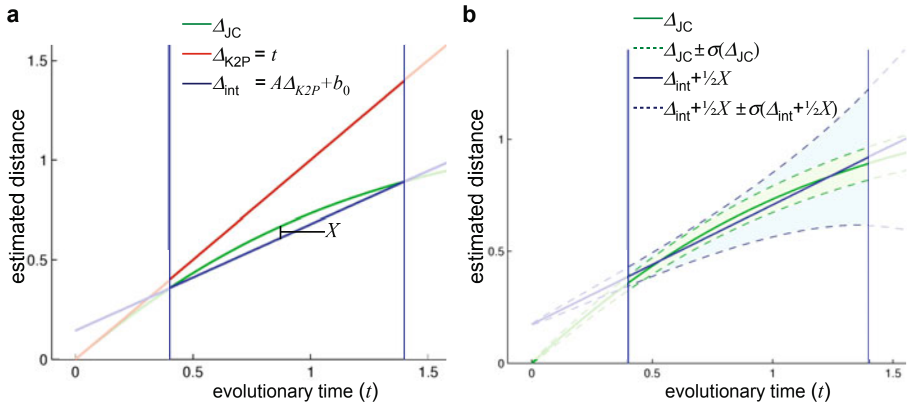


Fig. 1. Deviation from additivity and stochastic error. (a) Δ_{JC} is portrayed (green) in the homogeneous sub-model of $\mathcal{M}_{\mathrm{K2P}}$ with $R = 10$ in the interval $t \in [0.4, 1.4]$. Its linear interpolation in that interval, $\Delta_{\mathrm{int}} = At + b_0$, is plotted in blue, and the maximum difference between the two functions is designated by X. The deviation of Δ_{JC} from additivity within this setting can be shown to be $\frac{X}{2A}$ (A being the slope of Δ_{int}). **(b)** The affine-additive SR function minimizing its deviation from Δ_{JC} is $\Delta_{\mathrm{int}} + \frac{1}{2}X$. The stochastic error margins for the two SR functions, assuming sequence length of 500 bp, are indicated by the area between dashed lines.

4 Performance of Non-additive SR Functions in Quartet Resolution

In this section we address the specific task of quartet reconstruction. We assume that the true model tree is a homogeneous K2P quartet with ti-tv ratio $R > \frac{1}{2}$, and compare the performance of the non-additive function Δ_{JC} with the performance of Δ_{K2P} (or an affine transformation of it). The topology of a quartet spanning four taxa $\{1, 2, 3, 4\}$ is represented by split notation $(ij|kl)$ (where $\{i, j, k, l\} = \{1, 2, 3, 4\}$), indicating that the internal edge of the quartet separates i, j from k, l. The four-point method (FPM) [23,5] resolves this split using the six observed pairwise distances $\{\widehat{d}_{ij} : \{i, j\} \subset \{1, 2, 3, 4\}\}$: it first partitions the six observed distances into three sums $\widehat{d}_{12} + \widehat{d}_{34}$, $\widehat{d}_{13} + \widehat{d}_{24}$, and

$\widehat{d}_{14} + \widehat{d}_{23}$, and then determines the quartet split according to the minimal sum (the sum $\widehat{d}_{ij} + \widehat{d}_{kl}$ corresponds to the split $(ij|kl)$).

Figure 2 demonstrates the effect of the concavity of Δ_{JC} on the accuracy of the FPM on two types of quartets. Both types have an internal edge with length t_i, two long external edges with length t_l, and two short external edges with length t_s. In both types the quartet split is (12|34). In type A quartets (Fig. 2a), the short edges are on one side of the split and the long edges are on the other side. In this case, the sum associated with the split $(d_{12} + d_{34})$ is of the smallest and largest inter-leaf distances. The concavity of Δ_{JC} increases the separation between this sum and the other two competing sums, leading to an *improvement* in reconstruction accuracy. The other quartet configuration (type B; Fig. 2b) has a short edge and a long edge on both sides of the split. In this case, the interval of interpolation is $[d_{13}, d_{24}]$, and the distance $d_{12} = d_{34}$ is in the center of this interval. Thus the concavity of Δ_{JC} decreases the separation between the sums $d_{13} + d_{24}$ and $d_{12} + d_{34}$ by approximately twice the deviation from additivity of Δ_{JC} in that range. When the deviation from additivity exceeds $\frac{1}{2}t_i$, the sum $d_{13} + d_{24}$ becomes the minimal sum, and Δ_{JC} becomes inconsistent. Type B quartets, which are sometimes termed *Felsenstein quartets*, provide a worst case scenario for quartet resolution by a concave SR function.

Interestingly, Δ_{JC} ends up performing better than Δ_{K2P} even on many of these "worst case" quartets, since its smaller stochastic noise compensates for

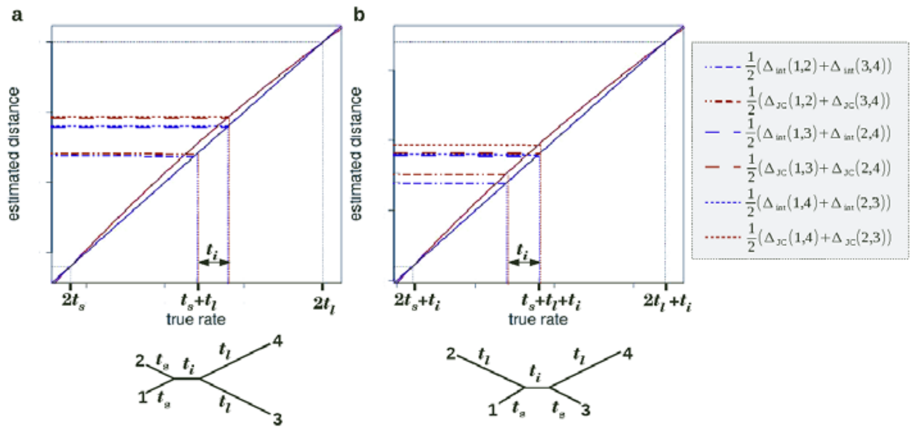


Fig. 2. Performance of the Four Point Method using Δ_{JC} on K2P quartets with ti-tv ratio $R = 2$. The concave non-additive SR function Δ_{JC} is shown (solid red line) in the interval $[t_0, t_1]$, where t_0 and t_1 are the smallest and largest of the six pairwise distances (resp.). The solid blue line shows the linear interpolation $\Delta_{int} = At + b_0$ of Δ_{JC} in the interval $[t_0, t_1]$. Horizontal bars correspond to half of each of the three sums computed by FPM under the two SR functions (see legend to the right). **(a)** In quartets of type A, the deviation from additivity of Δ_{JC} *increases* its FPM separation, compared to that of Δ_{int}. **(b)** In quartets of type B, the deviation from additivity of Δ_{JC} *decreases* its FPM separation, compared to that of Δ_{int}.

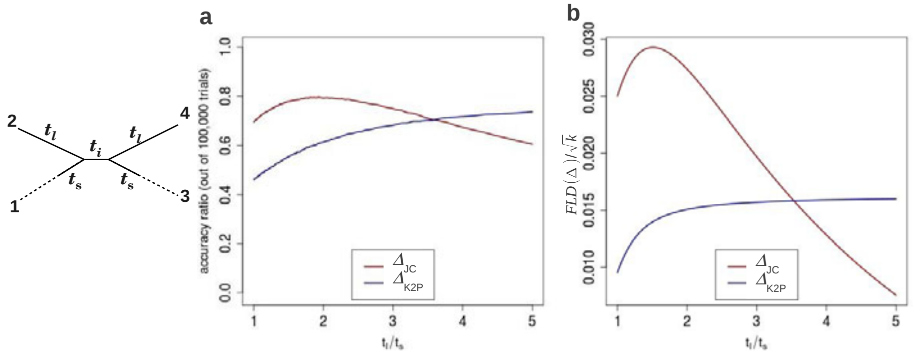


Fig. 3. Performance of Δ_{JC} and Δ_{K2P} on a series of quartets of type B. A series of homogeneous K2P quartets is considered (left illustration), with ti-tv ratio of $R = 5$, and edge rates $t_i = 0.2$, $t_l = 1$, and $t_s \in [0.2, 1]$. **(a)** Reconstruction accuracy using FPM and either Δ_{JC} (red) or Δ_{K2P} (blue) plotted against t_l/t_s. Accuracy ratio is estimated using 100,000 independent replicates for each parameter setting and sequences of length 1000 bp. **(b)** Fisher's Linear Discriminant (FLD) for the sums corresponding to splits (12|34) and (13|24) under either Δ_{JC} (red) or Δ_{K2P} (blue) plotted against t_l/t_s. For each SR function $\Delta \in \{\Delta_{JC}, \Delta_{K2P}\}$, we plot $FLD(\Delta)/\sqrt{k}$, where k is the sequence length.

the deviation from additivity (see also Fig. 1). This is demonstrated in the experiment described in Figure 3a, where series of homogeneous K2P quartets of type B are considered. For each quartet in the series, we generated a total of 100,000 simulations using 1000 bp long sequences, and we recorded the number of times (out of 100,000) it was accurately resolved using FPM and each of the two SR function (Δ_{JC} and Δ_{K2P}). Despite its deviation from additivity, Δ_{JC} outperforms the additive SR function Δ_{K2P} on many of these quartets. Only when the deviation from additivity is sufficiently large ($t_l/t_s > 3.6$ in these experiment), Δ_{K2P} outperforms Δ_{JC}. We note that as the sequence length grows, both methods become more accurate, but their relative behavior remains pretty much the same, with a similar crossover point around 3.6 (results not shown).

4.1 Using Fisher's Linear Discriminant

In order to provide a better understanding (and ability to predict) the results of similar experiments, we present a simple and general framework based on Fisher's linear discriminant (FLD). FLD measures the separation between normal random variables $X \sim N(\mu_1, \sigma_1)$ and $Y \sim N(\mu_2, \sigma_2)$ using the following measure ([7,2]):

$$FLD(X,Y) = \frac{|\mu_1 - \mu_2|}{\sqrt{\sigma_1^2 + \sigma_2^2}} . \tag{6}$$

We use FLD to measure the separability of the distance sum corresponding to the true split (which should be the minimal sum for consistent SR functions) from the two remaining sums. For the expectation μ of each sum we use the true distances as computed by the SR function on the actual model parameters. For the variance σ^2, we use the sum of the approximate variances of the two distances involved in the sum. We expect that an SR function which provides a larger separation of the smallest sum from the two other sums will imply a better reconstruction probability. Figure 3b plots FLD of Δ_{JC} and Δ_{K2P} associated with the comparison of the true split (12|34) and the "Δ_{JC} favored split" (13|24) along the quartet series considered in Figure 3a. Since both expressions are proportional to square-root of the sequence length ($\sqrt{k}$), the comparison of the two SR functions using FLD is independent of sequence length. We designate this by plotting $FLD(\Delta)/\sqrt{k}$. As shown, the equilibrium point of the Fisher discriminants of Δ_{JC} and Δ_{K2P} is pretty close to the equilibrium point of the accuracy of reconstructions of these two functions. This is despite the fact that the formal conditions required from FLD (namely that the two values would be independent normal variables) are not strictly met.

Perhaps the most useful feature of this framework is the natural way in which it teases apart the stochastic error from the deviation from additivity. If we denote the enumerator of FLD by SEP and its denominator by $NOISE$, then a comparison of FLD estimates between two SR function Δ_1, Δ_2 can be represented as a ratio of ratios:

$$\frac{FLD(\Delta_1)}{FLD(\Delta_2)} = \frac{SEP(\Delta_1)}{SEP(\Delta_2)} \bigg/ \frac{NOISE(\Delta_1)}{NOISE(\Delta_2)} \, . \tag{7}$$

In Figure 4 we compare the SEP and $NOISE$ ratios for Δ_{JC} and Δ_{K2P} for the series of type B quartets assumed in the previous experiment, with ti-tv ratio

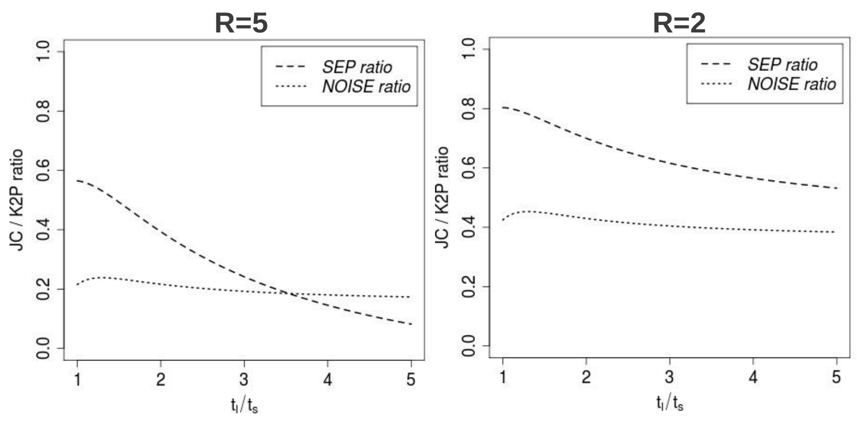


Fig. 4. *SEP* **and** *NOISE* **ratios.** $SEP(\Delta_{\mathrm{JC}})/SEP(\Delta_{\mathrm{K2P}})$ (dashed) and $NOISE(\Delta_{\mathrm{JC}})/NOISE(\Delta_{\mathrm{K2P}})$ (dotted) plotted against t_l/t_s for the series of homogeneous K2P quartets of type B assumed in Fig. 3, with ti-tv ratio of $R = 5$ (left) and $R = 2$ (right).

of $R = 5$ (left) and ti-tv ratio of $R = 2$ (right). These plots indicate that the $NOISE$ ratios are constant throughout this series. This is because the stochastic noise is dominated by the longest path in the quartet (the one connecting taxa 2 and 4), and that path does not change throughout the series. Conversely, the SEP ratio is reduced as the ratio t_l/t_s increases, due to an increase in the deviation of Δ_{JC} from additivity. This reduction in the SEP ratio becomes more dramatic as the ti-tv ratio grows. More examples of this type of analysis using the FLD are presented in [4].

5 Inferring Trees from Genomic Sequences

The previous section demonstrates the usefulness of non-additive SR functions in a controlled setting where the trees are quartets and the true substitution model is simple and known. In order to test the usefulness of this approach in more realistic scenarios we conducted several experiments on data simulated on larger trees, as well as experiments on real genomic data. A complete description of these experiments is presented in [4]. In this section we summarize the main findings from our experiment on genomic data.

Our genomic data set is based on a set of 31 *clusters of orthologous groups* (COGs) compiled by Ciccarelli et al. [3]. These gene families were selected to capture the evolutionary history of the species from which they are extracted. We identified 163 bacterial species which contained high quality alignments across the 31 gene families. The 31 alignments were concatenated to form one long 163-way multiple sequence DNA alignment. The full technical details on the procedure of obtaining this alignment are provided in [4]. As a reference tree for comparing the reconstructed trees, we used the phylogenetic tree of microbial species provided by the Living Tree Project [22]. This tree, spanning 8,029 species at the time of writing, is based on widely accepted analysis of the small subunit (SSU) 16S RNA. Although it is based on a very small genomic region, its careful curation makes it a reasonable proxy for the true species tree. A subtree spanning our 163 bacterial species was extracted from this tree and treated as the true phylogenetic tree in our analysis.

We used the base set of 163 species to generate 40,000 random 10-species sub-alignments. In the subsequent reconstruction process only four-fold degenerate sites of these sub-alignments were used, omitting those that contained gap symbols. Each sub-alignment was used to compute three distance matrices – one under Δ_{JC}, one under Δ_{K2P}, and one under the LogDet SR function Δ_{LogDet}. The LogDet function [19,15] is used here since it is additive in the general time-reversible model, which is a general substitution model containing many common models, such as K2P. The neighbor joining algorithm (NJ) [17,21] was then applied to the three matrices and the resulting trees were compared to the reference LTP tree according to the Robinson-Foulds (RF) distance. For comparison, we used a fourth reconstruction method that is a-priori expected to be better suited than the other three in dealing with biological sequences. This reconstruction method (termed BIONJ-GTR) uses the BIONJ reconstruction algorithm [8] on distances obtained under the general time-reversible model with invariant sites and Gamma distribution of rates across sites (GTR+Γ+I) [14,16]. We use the PhyML package [11] to infer this tree for each subset.

Our results show that in 83% of the 40,000 subsets, the tree inferred using Δ_{JC} had an equal or lower RF-distance to the reference tree than the tree reconstructed by BIONJ-GTR. Moreover, Δ_{LogDet} was, by far, the least accurate of all methods. These findings endorse our main argument that distances corresponding to better-fitting models can lead to less accurate reconstruction. In order to sort out the results obtained across the 40,000 subsets, we partitioned them according to the RF-distance of the BIONJ-GTR tree from the reference tree. Results are shown in Figure 5. The mean accuracy of Δ_{JC} is higher than that of Δ_{K2P} and Δ_{LogDet} across all partitions. The advantage of Δ_{JC} over the other two methods seems to slightly increase with the accuracy of the BIONJ-GTR tree. This indicates that over-simplified distance estimation techniques are especially beneficial when the sequence data conveys a stronger phylogenetic signal.

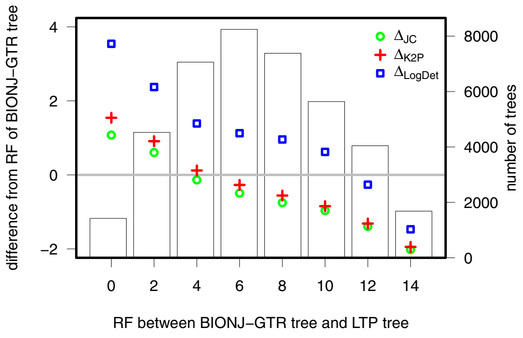


Fig. 5. Evaluation against BIONJ-GTR tree. The 40,000 subsets of size 10 were partitioned according to the the RF-distance of the tree reconstructed using BIONJ-GTR from the reference LTP tree (X axis). The Y axis describes the difference between the RF-distance associated with a particular SR function (Δ_{K2P}, Δ_{JC}, or Δ_{LogDet}) and the RF-distance associated with BIONJ-GTR. The bar plot in the background depicts the number of subsets in each partition.

6 Discussion and Outlook

In this paper we study basic properties of evolutionary distance estimation using SR functions and how they affect the accuracy of phylogenetic reconstruction. When studying accuracy of statistical estimates, it is important to consider both the bias of the estimate and its variance (often referred to as stochastic noise). In some cases it might be worth trading off variance for bias, resulting in a slightly skewed estimate which is less noisy. A challenge in carrying out such a study for statistical estimation of evolutionary distances is that bias is not completely well-defined in this case, since the "true" evolutionary distances can take many forms; any affine-additive SR function is a valid one for the purpose of phylogenetic reconstruction.

We introduce the concept of *deviation from additivity* to quantify the bias of an SR function in a homogeneous substitution model. We demonstrate this analytic framework by studying the bias of the Jukes-Cantor SR function (Δ_{JC}) in Kimura's two parameter model when the ti-tv ratio is strictly greater than $\frac{1}{2}$. We show that even when the ti-tv ratio is as high as 5 or 10, this bias is small enough such that the reduced variance of Δ_{JC} makes it overall more accurate than Kimura's SR function (Δ_{K2P}), which has no bias. We show this using analytic bounds as well as detailed simulation experiments on quartet trees.

These results were also affirmed in a round of experiments on real biological sequences. In the case of real data, the true substitution model is likely to be very complex, and all common distance formulas are expected to have some bias. Our results show that simpler SR functions with lower variance lead to more accurately reconstructed trees on average, compared to SR functions that are expected to have reduced bias but higher variance.

With the devised framework at hand, the study of distance estimation can be extended in different directions. More complex models and non-additive SR functions could be studied, and improved methods for the analysis of biological sequences could be established. Additionally, there is a need for extending the FLD-based heuristic to trees larger than quartets. Finally, incorporating our methods in existing software for phylogenetic reconstruction looks like a promising venue for increasing the accuracy of distance-based phylogenetic reconstruction at low (or even negative) computational cost.

References

1. Atteson, K.: The performance of neighbor-joining methods of phylogenetic reconstruction. Algorithmica 25, 251–278 (1999)
2. Bishop, C.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
3. Ciccarelli, F.D., Doerks, T., von Mering, C., Creevey, C.J., Snel, B., Bork, P.: Toward automatic reconstruction of a highly resolved tree of life. Science 311(5765), 1283–1287 (2006)
4. Doerr, D., Gronau, I., Moran, S., Yavneh, I.: Stochastic errors vs. modeling errors in distance based phylogenetic reconstructions (2011) (in preparation), http://www.cs.technion.ac.il/~moran/r/wabi-in-prep.pdf
5. Erdos, P., Steel, M., Szekely, L., Warnow, T.: A few logs suffice to build (almost) all trees (I). Random Structures Algorithms 14, 153–184 (1999)
6. Felstenstein, J., Sober, E.: Parsimony and likelihood: an exchange. Systematic Zoology 35, 617–626 (1986)
7. Fisher, R.: The use of multiple measurements in taxonomic problems. Annals of Eugenics 7, 177–188 (1936)
8. Gascuel, O.: BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. Mol. Biol. Evol. 14(7), 685–695 (1997)
9. Gronau, I., Moran, S., Yavneh, I.: Towards optimal distance functions for stochastic substitution models. J. Theor. Biol. 260(2), 294–307 (2009)

10. Gronau, I., Moran, S., Yavneh, I.: Adaptive distance measures for resolving K2P quartets: Metric separation versus stochastic noise. J. Comp. Biol. 17(11), 1391–1400 (2010)
11. Guindon, S., Gascuel, O.: A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. Systematic Biology 52, 696–704 (2003)
12. Jukes, T., Cantor, C.: Evolution of protein molecules. In: Munro, H. (ed.) Mammalian Protein Metabolism, pp. 21–132. Academic Press, New York (1969)
13. Kimura, M.: A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. J. Mol. Evol. 16(2), 111–120 (1980)
14. Lanave, C., Preparata, G., Saccone, C., Serio, G.: A new method for calculating evolutionary substitution rates. J. Mol. Evol. 20, 86–93 (1984)
15. Lockhart, P., Steel, M., Hendy, M., Penny, D.: Recovering evolutionary trees under a more realistic model of sequence evolution. Mol. Biol. Evol. 11(4), 605–612 (1994)
16. Rodriguez, F., Oliver, J.L., Marin, A., Medina, J.R.: The general stochastic model of nucleotide substitution. J. Theor. Biol. 142, 485–501 (1990)
17. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 4, 406–425 (1987)
18. Sober, E.: A likelihood justification of parsimony. Cladistics 1, 209–233 (1985)
19. Steel, M.: Recovering a tree from the leaf colourations it generates under a Markov model. Appl. Math. Lett. 7(2), 19–24 (1994)
20. Steel, M., Penny, D.: Parsimony, likelihood, and the role of models in molecular phylogenetics. Mol. Biol. Evol. 17, 839–850 (2000)
21. Studier, J., Keppler, K.: A note on the neighbor-joining algorithm of Saitou and Nei. Mol. Biol. Evol. 5(6), 729–731 (1988)
22. Yarza, P., Ludwig, W., Euzeby, J., Amann, R., Schleifer, K.H., Glockner, F.O., Rossello-Mora, R.: Update of the All-Species Living Tree Project based on 16S and 23S rRNA sequence analyses. Syst. Appl. Microbiol. 33, 291–299 (2010)
23. Zaretskii, K.: Constructing a tree on the basis of a set of distances between the hanging vertices. Uspekhi Mat Nauk 20(6), 90–92 (1965) (in Russian)

Constructing Large Conservative Supertrees*

Jianrong Dong and David Fernández-Baca

Department of Computer Science, Iowa State University, Ames, IA 50011, USA

Abstract. Generalizations of the strict and loose consensus methods to the supertree setting, recently introduced by McMorris and Wilkinson, are studied. The supertrees these methods produce are conservative in the sense that they only preserve information (in the form of splits) that is supported by at least one the input trees and that is not contradicted by any of the input trees. Alternative, equivalent, formulations of these supertrees are developed. These are used to prove the NP-completeness of the underlying optimization problems and to give exact integer linear programming solutions. For larger data sets, a divide and conquer approach is adopted, based on the structural properties of these supertrees. Experiments show that it is feasible to solve problems with several hundred taxa and several hundred trees in a reasonable amount of time.

1 Introduction

Supertrees synthesize the information from a collection of phylogenetic trees for different partially overlapping sets of taxa — the input trees — into a comprehensive phylogeny for all the species — a supertree. Beginning with the paper by Gordon where supertrees were introduced [13], a variety of supertree construction methods have been proposed (for surveys, see [4,18]). We cannot discuss supertree construction without referring to matrix representation with parsimony (MRP) [2], by far the most commonly used method. While MRP often performs well [5], it is essentially an ad hoc adaptation of parsimony to the supertree setting. This is reflected in the fact that MRP supertrees may display relationships that are not supported by any of the input trees [12].

Here we consider methods based on *splits*, where a split is the bipartition of the taxon set induced the removal of an edge from a tree (see Section 2). Following [22], we say that a supertree method is *liberal* if it allows the inclusion of splits that are contradicted by some subset of the input trees, provided a certain optimization criterion is met. *Conservative* methods produce supertrees that only display splits that are not contradicted by any input tree. Previous work [8,9,10,11] has characterized the mathematical and computational properties of majority-rule supertrees, a liberal approach. Recently, McMorris and Wilkinson introduced two conservative methods, strict and loose supertrees [15], which generalize strict and loose consensus trees.

In a loose supertree, each split is compatible with every input tree and is supported by at least one input tree. Each split in a strict supertree must meet the same requirements as in a loose supertree, plus one additional property: Every input tree either supports the

* This work was supported in part by the National Science Foundation under grants DEB-0829674 and CCF-1017189.

split or it has no say on it, because its leaf set does not have sufficient overlap with both sides of the split (i.e., the split is trivial in the input tree — see Section 2). The formal definitions of strict and loose supertrees are more complex than this, since one must be careful in expressing the notions of support and conflict in the context of partial overlap among trees (see Section 3). One appealing property of strict and loose supertrees is that, in a well-defined sense, they express the least that we could expect from any split-based supertree method. Thus, they offer a baseline against which other methods can be compared. Furthermore, while the requirements imposed by their formulation might seem stringent, conservative supertrees can be quite well-resolved.

We provide alternative and equivalent definitions of loose and strict supertrees, based on Robinson-Foulds distance, along the lines of those proposed for majority-rule supertrees [8,10]. They rely on the idea of filling in the input trees with the taxa missing in them in such a way as to maximize the amount of agreement. There may be multiple ways to achieve this; the supertree returned is the strict consensus of all optimal solutions. Based on our formulations, we show that obtaining strict and loose conservative supertrees is NP-complete. The new definition also enables us to express supertree construction as an optimization problem based on Robinson-Foulds distance, which we can solve exactly using integer linear programming (ILP). This method can handle problems of moderate size, with at most 40 taxa and fewer than five trees. For larger data sets, we turn to heuristics.

The idea behind our heuristics is simple: If we were somehow able to guess any of the splits in the strict or loose supertree, we could break down the original problem into two subproblems, one for each side of the split. Since we do not know the splits in advance, we adopt a greedy approach and seek a split that at each step optimizes a function based on Robinson-Foulds distance. The computational experience reported here shows that the downsizing effect of divide and conquer can drastically reduce the running time, allowing us to solve problems with hundreds of taxa and trees.

While the above divide-and-conquer technique performs extremely well, it is still only a heuristic. To increase our confidence in the solution, we verify each of its edges, checking to see if there is a better tree that does not include it. We also attempt multiple runs with different decompositions. Even with all these safeguards, our supertrees may not be precisely *the* strict or loose supertrees originally defined by McMorris and Wilkinson. Nevertheless, the constraints we impose guarantee that our heuristically-built supertrees are conservative. That is, every loose supertree that we generate has the property that each of its splits is supported by some tree and contradicted by none. Also, every strict supertree we generate has the property that for each input tree, each of the splits of the supertree is either trivial in the tree or is supported by the tree.

Robinson-Foulds distance has been used to build supertrees in other work, including the aforementioned majority-rule supertrees, and in the rooted and unrooted Robinson-Foulds supertrees of [1,7]. None of these methods can be called conservative. Indeed, for technical reasons, the Robinson-Foulds supertrees of [1,7] are required to be fully resolved, which guarantees many unsupported splits. Closer to the spirit of the methods studied here is PhySIC [16]. While PhySIC is based on triplets, not splits, and thus assumes rooted input trees, its underlying principles ensure conservativeness.

There is some debate about the right way to measure the quality of a supertree. Swenson et al. [21] argue that evaluating supertree methods solely on how accurately they reflect the input trees can be problematic, as it may not say that much on how faithfully they reconstruct the true tree. While these conclusions are based on simulations under a particular set of experimental conditions, they do raise significant issues for methods such as the ones studied here. Perhaps it is better to view conservative supertrees as a way to summarize phylogenetic data in a principled manner rather than as a method to somehow compensate for errors in the input trees. In our opinion, there is room for both approaches in a phylogenetic toolbox.

2 Preliminaries

A *phylogenetic tree* is an unrooted leaf-labeled tree where every internal node has degree at least three. We will use "tree" and "phylogenetic tree" interchangeably. The leaf set of a tree T is denoted by $\mathscr{L}(T)$. Each leaf is called a *taxon* (plural, *taxa*).

A *profile* is a tuple of trees $P = (t_1, \ldots, t_k)$. Each t_i in P is called an *input tree*. Let $\mathscr{L}(P) = \bigcup_{i=1}^{k} \mathscr{L}(t_i)$ and $n = |\mathscr{L}(P)|$. Tree T is a *supertree* for P if $\mathscr{L}(T) = \mathscr{L}(P)$.

A *split* is a bipartition of a set. The split whose parts are A and B is denoted $A|B$. The order here does not matter, so $A|B$ is the same as $B|A$. Split $A|B$ is *nontrivial* if each of A and B has at least two elements; otherwise it is *trivial*. Split $A|B$ *extends* another split $C|D$ if $A \supseteq C$ and $B \supseteq D$, or $A \supseteq D$ and $B \supseteq C$.

Tree T *displays* split $A|B$ if there is an edge in T whose removal gives trees T_1 and T_2 such that $A \subseteq \mathscr{L}(T_1)$ and $B \subseteq \mathscr{L}(T_2)$. A split $A|B$ is *full* with respect to a tree T if $A \cup B = \mathscr{L}(T)$.

The set of all nontrivial full splits displayed by T is denoted $\mathrm{Spl}(T)$. It is well known that the full splits of T uniquely identify T [19, p. 44]. Thus, we will often view trees as sets of splits and write "$A|B \in T$" if tree T displays split $A|B$. A tree T with $\mathrm{Spl}(T) = \emptyset$ is called a *fan*.

Let $S \subseteq \mathscr{L}(T)$. The *restriction of T to S*, denoted $T|_S$, is the phylogenetic tree with leaf set S such that

$$\mathrm{Spl}(T|_S) = \{A \cap S | B \cap S : A|B \in \mathrm{Spl}(T) \text{ and } |A \cap S|, |B \cap S| > 1\}.$$

Let T' be a phylogenetic tree such that $S = \mathscr{L}(T') \subseteq \mathscr{L}(T)$. Then, T *displays* T' if $\mathrm{Spl}(T') \subseteq \mathrm{Spl}(T|_S)$. Trees T_1 and T_2 are *compatible* if there exists a tree T such that T displays T_1 and T_2.

A set of splits is *compatible* if there is a tree T that displays them all. Tree T is *compatible with* a set of splits $\mathscr{X}$ if there is a tree T' that displays T and $\mathscr{X}$.

Let t be an input tree in a profile P, T be a supertree for P, and $A|B \in T$. Let $A' = A \cap \mathscr{L}(t)$ and $B' = B \cap \mathscr{L}(t)$. We say that t *supports* $A|B$ if $A'|B' \in t$; t is *in conflict with* $A|B$ if $A'|B'$ is incompatible with t.

Let T_1 and T_2 be two phylogenetic trees over the same leaf set. The *symmetric-difference distance*, also known as *Robinson-Foulds distance* [17], between T_1 and T_2, denoted $d(T_1, T_2)$, is defined as

$$d(T_1, T_2) = |(\mathrm{Spl}(T_1) \setminus \mathrm{Spl}(T_2)) \cup (\mathrm{Spl}(T_2) \setminus \mathrm{Spl}(T_1))|.$$

The *asymmetric difference* from T_1 to T_2, denoted $AD(T_1,T_2)$, is defined as

$$AD(T_1,T_2) = |(\mathrm{Spl}(T_1) \setminus \mathrm{Spl}(T_2))|$$

Rooted phylogenetic trees can be viewed as a special case of unrooted trees. That is, we can view a profile of rooted trees as unrooted trees, all of which have a common taxon called the root. Thus, in a split in a rooted tree, one of the two parts must contain the root; the part that does not contain it is called a *cluster*.

The *consensus problem* is the special case of the supertree problem where the profile $P = (T_1,\ldots,T_k)$ consists of trees that have the same leaf set. The *strict consensus* of P, denoted $\mathrm{Str}(P)$ is the tree that displays exactly the full splits that are in every tree in the profile. The *loose consensus* of P, denoted $\mathrm{Loose}(P)$, is the tree that displays exactly the full splits that are in some tree of P and compatible with every tree in P.

3 Strict and Loose Supertrees

McMorris and Wilkinson [15] defined strict and loose supertrees using asymmetric distance as follows. Let $P = (t_1,\ldots,t_k)$ be a profile and S be a supertree for P. Define $AD^-(S,P) = \sum_{i=1}^{k} AD(S|L_i,t_i)$ and $AD^-(P,S) = \sum_{i=1}^{k} AD(t_i,S|L_i)$. Define the following two sets of candidate supertrees: $\sigma = \{S : AD^-(S,P) = 0 \text{ and } \mathscr{L}(S) = \mathscr{L}(P)\}$ and $\lambda = \{S : S \text{ is compatible with } P \text{ and } \mathscr{L}(S) = \mathscr{L}(P)\}$. The *strict supertree* for P, denoted $\mathrm{Str}_s(P)$, is the strict consensus of the supertrees in the set $\sigma^* = \{S \in \sigma : AD^-(P,S) \text{ is minimum}\}$. The *loose supertree* of P, denoted $\mathrm{Loose}_s(P)$, is the strict consensus of all trees in the set $\lambda^* = \{S \in \lambda : AD^-(P,S) \text{ is minimum}\}$. It can be shown that (i) each split in $\mathrm{Loose}_s(P)$ is supported by at least one tree in P and is compatible with every tree in P and (ii) for every split $A|B$ in $\mathrm{Str}_s(P)$ and each tree t in P, it must be the case that either t supports $A|B$ or $A|B$ is trivial in t.

We now develop alternative definitions of strict and loose supertrees based on symmetric distances and on two different notions of the *span* of a tree. Let t be a tree in a profile P. The *graft/refine span* of t is the set $\langle t\rangle_{gr}$ of supertrees of P that refine t; i.e., $\langle t\rangle_{gr} = \{T : T \text{ displays } t \text{ and } \mathscr{L}(T) = \mathscr{L}(P)\}$. The *graft-only span* of t is the set $\langle t\rangle_g$ of all supertrees of P obtained by adding leaves and splits to t without refining the original splits in t. That is, $\langle t\rangle_g = \{T : T|_{\mathscr{L}(t)} = t \text{ and } \mathscr{L}(T) = \mathscr{L}(P)\}$. Observe that, by definition, $\langle t\rangle_g \subseteq \langle t\rangle_{gr}$. In particular, in the consensus setting (where $\mathscr{L}(t) = \mathscr{L}(P)$), $\langle t\rangle_g = \{t\}$, while $\langle t\rangle_{gr}$ includes all the refinements of t. The *graft-only span* of a profile P is $\langle P\rangle_g = (T_1,\ldots,T_k)$ where $T_i \in \langle t\rangle_g$ for every i. The *graft/refine span* of a profile P is $\langle P\rangle_{gr} = (T_1,\ldots,T_k)$ where $T_i \in \langle t\rangle_{gr}$ for every i.

The *score* of a profile $R = (T_1,\ldots,T_k)$ of trees over the same leaf set, denoted $\mathrm{score}(R)$, is defined as $\mathrm{score}(R) = \sum_{i=1}^{k} d(\mathrm{Str}(R),T_i)$. Now, let P be an arbitrary profile. Let $\mathscr{R}^*$ be the set of all $R \in \langle P\rangle_g$ such that $\mathrm{score}(R)$ is minimum. For each $R \in \mathscr{R}^*$, we refer to $\mathrm{Str}(R)$ is an *optimal strict candidate supertree* (strict OCT) for P. An *optimal loose candidate supertree* (loose OCT) for P is defined analogously, with “strict” replaced by “loose” and “$\langle P\rangle_g$” replaced by “$\langle P\rangle_{gr}$”.

Theorem 1. *For any profile P,* $\mathrm{Str}_s(P)$ *is the strict consensus of all strict OCTs for P and* $\mathrm{Loose}_s(P)$ *is the strict consensus of all loose OCTs for P.*[1]

[1] Note the deliberate use of strict consensus for both strict and loose supertrees.

Theorem 1 gives an alternative perspective on strict and loose supertrees. The use of the graft-refine span in loose supertrees implies that $\mathrm{Loose}_s(P)$ can contain a split that is not supported by all the input trees, as long as it is supported by one tree and compatible with all the trees. On the other hand, the use of the graft-only span in strict supertrees implies that a split can appear in $\mathrm{Str}_s(P)$ only if, for each input tree, the split is either supported by that tree or it becomes trivial when reduced to the leaf set of that tree. These facts give an alternative justification to the observation that strict and loose supertrees *generalize* strict and loose consensus trees [15]. Formally:

Theorem 2. *For any profile P where the input trees have identical leaf sets,* $\mathrm{Str}_s(P) = \mathrm{Str}(P)$ *and* $\mathrm{Loose}_s(P) = \mathrm{Loose}(P)$.

Thus, we henceforth drop the subscript "s" from the notation for strict and loose supertrees and simply write "$\mathrm{Str}(P)$" and "$\mathrm{Loose}(P)$".

It follows from their definitions that the loose and strict supertrees of a profile are compatible with every input tree. Indeed, each split in the strict or loose supertree is supported by at least one input tree and in conflict with no input tree. Rather counterintuitively, however, there exist profiles for which the strict supertree contains splits not present in the loose supertree, something that is never true in the consensus setting.

The next result can be proved by reduction from the quartet compatibility problem, which is known to be NP-complete [20].

Theorem 3. *There is no polynomial-time algorithm to construct strict or loose OCTs unless $P = NP$.*

4 ILP Formulations

4.1 Formulations Based on the Restricted Span

Our first ILP formulation expresses the task of finding a strict or loose OCT (see Section 3) as an optimization problem.

The *restricted span of a tree t* in a profile P, denoted $\langle t \rangle_r$, is the set of all supertrees T for P such that every nontrivial split in T extends a distinct nontrivial split in t [10]. Note that $\langle t \rangle_r \subseteq \langle t \rangle_g \subseteq \langle t \rangle_{gr}$ and that if $T \in \langle t \rangle_r$, then $|\,\mathrm{Spl}(T)| = |\,\mathrm{Spl}(t)|$. The *restricted span of a profile* $P = (t_1, \ldots, t_k)$, denoted $\langle P \rangle_r$ is the set of all profiles $R = (T_1, \ldots, T_k)$ such that $T_i \in \langle t_i \rangle_r$ for $i \in \{1, \ldots, k\}$. As shown in [10], the set of all profiles in the restricted span of P can be nicely expressed using integer linear constraints. We now summarize the main ideas.

For each $j \in \{1, \ldots, k\}$, let m_j be the number of nontrivial splits in t_j. A *matrix representation* of t_j is a $n \times m_j$ matrix $M(t_j)$ whose columns correspond to the nontrivial splits of t_j. Suppose column i of $M(t_j)$ corresponds to split $A|B$ in t_j and let x be a taxon in $\mathscr{L}(P)$. Then, $M_{x,i}(t_j) = 1$ if $x \in A$, $M_{x,i}(t_j) = 0$ if $x \in B$, and $M_{x,i}(t_j) =?$ otherwise. Let $m = \sum_{j=1}^{k} m_j$. A *matrix representation* of P, denoted $M(P)$, is a $n \times m$ matrix $M(P)$ obtained by concatenating matrices $M(t_1)$ through $M(t_k)$.

A *legal fill-in* of $M(P)$ is a matrix M' obtained by replacing each "?" in $M(P)$ with a "1" or a "0", in such a way that $M' = M(R)$ for some $R \in \langle P \rangle_r$. Clearly, for every $R \in \langle P \rangle_r$ there is a legal fill-in corresponding to R. For each row x and column i, create a binary *fill-in variable* F_{xi} if $M_{xi}(P) = ?$. It is straightforward to write integer linear constraints to ensure the fill-in variables are assigned values that yield a legal fill-in of $M(P)$. The main function of these constraints is to avoid the "four gametes condition", so that, for each $i \in \{1,\ldots,k\}$, the filled-in submatrix for input tree t_i corresponds to a tree in $\langle t_i \rangle_r$ (for details, see [10]). The size of the ILP is $O(nm^2)$.

To obtain strict and loose supertrees from the restricted span, we use a variation on a technique from [10]. The first observation is that it suffices to have an ILP for finding *one* loose or strict OCT. A *verifier* is a procedure that takes a loose or strict OCT T and any split $A|B \in T$ and determines if there is another OCT T' that *does not* contain $A|B$. By Theorem 1, each split in Loose(P) (Str(P)) must be in every loose (strict) OCT T. Thus, if the verifier answers "yes" when given an OCT T and some split $A|B \in T$, $A|B$ is *not* in Loose(P) (Str(P)). Therefore, by repeatedly applying the verifier on an OCT, discarding any splits that do not pass the test, we are left with the loose (strict) supertree. As discussed in [10], given an ILP for finding an OCT, we can implement a verifier by putting $O(mn)$ additional variables and constraints on the original ILP requiring that $A|B$ is not displayed. If the resulting ILP has a higher objective value, then $A|B \in$ Loose(P) (Str(P)); otherwise, $A|B \notin$ Loose(P) (Str(P)).

We now describe ILPs for finding loose and strict OCTs based on the restricted span. The key idea is to add implicitly certain splits to the profiles $R \in \langle P \rangle_r$. Let us consider loose OCTs first. Suppose that $R = (T_1,\ldots,T_k)$ is a profile of trees over the same leaf set. Then, the *loose completion* of R is the profile $R' = (T_1',\ldots,T_k') \in \langle R \rangle_{gr}$, where $T_i' = T_i \cup \text{Loose}(R)$. Let $\mathscr{R}_L^*(P)$ be the set of all profiles G such that G is the loose completion of some $R \in \langle P \rangle_r$ and score(G) is minimum. The proof of the next result is along the lines of those of Theorems 3 and 4 of [10].

Theorem 4. *Let P be an arbitrary profile. Then for every $G \in \mathscr{R}_L^*(P)$,* Str(G) *is a loose OCT.*

Let G be the loose completion of some $R \in \langle P \rangle_r$. Observe that score$(G)$ equals the sum over all trees t in R of the number of splits in t that are not in Loose(R). The reason is that all loose splits of R are in every tree in G and, thus, do not contribute to the total distance from G to Str(G). This observation allows us to compute score(G) directly from R, without constructing G explicitly. To do this, we add to our ILP a binary variable w_i for each column i of $M(P)$, and create additional variables and constraints so that, for a given fill-in of $M(P)$, we have $w_i = 1$ if and only if the corresponding filled-in column (split) is compatible with every other column. I.e., $w_i = 1$ if and only if the resulting split is in the loose consensus. The objective function becomes $\sum_{i=1}^{m}(1 - w_i)$.

For strict supertrees, let $RL(R)$ be the set of all splits $A|B$ of $\mathscr{L}(P)$, such that $A|B \in$ Loose(R) and, for every tree t in the original profile P, either $A|B$ is supported by t or $A|B$ is trivial when restricted to $\mathscr{L}(t)$ The *RL-completion* of $R = (T_1,\ldots,T_k) \in \langle P \rangle_r$ is $R' = (T_1',\ldots,T_k')$ where $T_i' = T_i \cup RL(R)$. Let $\mathscr{R}_S^*(P)$ be the set of all profiles G such that G is the RL-completion of some $R \in \langle P \rangle_r$ and score(G) is minimum. Similarly to Theorem 4, we have the following.

Theorem 5. *Let P be an arbitrary profile. Then for every $G \in \mathscr{R}_S^*(P)$,* Str(G) *is a strict OCT.*

In analogy to loose supertrees, for any $R \in \langle P \rangle_r$, the value of score$(G)$ for the *RL*-completion G of R can be found without constructing G explicitly. The only change with respect to the loose case is that now the splits present in G but not in R must not only be in Loose(R), but they must also be supported by or be trivial in every underlying input tree. This condition can again be expressed with integer linear constraints.

4.2 ILPs for Finding Good Splits

The asymmetric distance-based formulations of conservative supertrees (see [15] and Section 3) can also be expressed using integer linear programming. This approach tends to yield larger problems than the one based in restricted spans, because the feasible solutions represent trees whose splits are completely unknown. Nevertheless, restricted versions of this problem can be very useful in finding splits that allow us to decompose the original problem into smaller subproblems. We consider two problems.

1. Find a nontrivial split $A|B$ of $\mathscr{L}(P)$ such that (i) for every input tree t, $A|B$ is either supported by t or is trivial in t, and (ii) the tree S whose only nontrivial split is $A|B$ minimizes $AD^-(P,S)$.
2. Find a nontrivial split $A|B$ of $\mathscr{L}(P)$ that is compatible with every input tree and such that the tree S whose only nontrivial split is $A|B$ minimizes $AD^-(P,S)$.

Problem 1 is used in computing strict supertrees; problem 2 helps in finding loose supertrees. We refer to a split $A|B$ that answers problems 1 or 2 as *good*. In Section 5, we discuss how a good split $A|B$ can be used to break down a conservative supertree problem on a profile P into two subproblems, one for the restriction of P to A, the other for the restriction of P to B. While there is no guarantee that $A|B$ is part of a strict or loose OCT, in practice, this approach gives an excellent heuristic. Furthermore, if there is no good split, then we can be certain that the strict (or loose) supertree is a fan.

To formulate the problem of finding a good split, define a binary vector $y = (y_1, \dots, y_n)$, indexed by $\mathscr{L}(P)$. Each assignment of values to the entries of y corresponds to a split of $\mathscr{L}(P)$, with $y_i = y_j$ if and only if taxa i and j are on the same side of the split. Constraints are needed to guarantee nontriviality; i.e., that y must contain at least two 0's and two 1's. Each input tree t_i is represented via the matrix $M(t_i)$ defined in Section 4.1.

The conditions on split $A|B$ imposed by problems 1 and 2 can easily be formulated as integer linear constraints. For every $i \in \{1, \dots, k\}$ we need a binary variable x_i, associated with input tree t_i, that is 1 if and only if $(A \cap \mathscr{L}(t_i))|(B \cap \mathscr{L}(t_i)) \in t_i$. This information is used to evaluate the objective function (given by asymmetric distance). The value of x_i can be expressed using integer linear constraints in terms of y and $M(t_i)$.

5 A Divide-and-Conquer Heuristic

The ILP formulation of Section 4.1 is exact, but is limited in the range of problems it can handle (see Section 6). We have found that good splits (in the sense of Section 4.2) yield an excellent heuristic approach to break the input profile into manageable subproblems. The same scheme, detailed in Algorithm 1, applies to both strict and loose supertrees.

Algorithm 1. DIVIDEANDCONQUER(P, Avoid)

```
1: if ($P$, Avoid) is sufficiently small then
2:     Find an OCT $T$ for ($P$, Avoid) using the span-based ILP of Section 4.1
3:     Let $z$ be the objective value of $T$
4:     return $[T, z]$
5: Search for a good split $A|B$ for ($P$, Avoid) using the method of Section 4.2
6: if no such split exists then
7:     Let $T$ be a fan on $\mathscr{L}(P)$
8:     Let $z = \sum\{|\,\mathrm{Spl}(t)| : t$ is an input tree in $P\}$
9:     return $[T, z]$
10: Create subproblems $(P_A, \mathsf{Avoid}_A)$ and $(P_B, \mathsf{Avoid}_B)$
11: $[T_A, z_A]$ = DIVIDEANDCONQUER$(P_A, \mathsf{Avoid}_A)$
12: $[T_B, z_B]$ = DIVIDEANDCONQUER$(P_B, \mathsf{Avoid}_B)$
13: Combine $T_B$ and $T_B$ to create $T$
14: $z = z_A + z_B$
15: return $[T, z]$
```

The input to Algorithm 1 is a profile P and a set Avoid of splits. The goal of the algorithm is to find a supertree T for P that does not display any of the splits in Avoid. The algorithm also returns T's objective value, z. Initially, Avoid is empty, but as the algorithm progresses, splits are added to the set. The threshold below which a problem is considered "small enough" in step 1 is determined empirically, as described in Section 6. As mentioned in Section 4.2, if step 5 fails to find a good split, the strict or loose supertree must be a fan.

The split $A|B$ of step 5 is computed by solving problem 1 or 2 of Section 4.2, depending on whether we are after strict or loose OCTs. If $A|B$ is found, it is used in step 10 to create the subproblems (P_A, Avoid_A) and (P_B, Avoid_B), to be handled recursively. The subproblems are constructed by restricting (P, Avoid) to A and B as follows. For each split $C|D \in \mathsf{Avoid}$, we add split $(C \cap A)|(D \cap A)$ to Avoid_A and $(C \cap B)|(D \cap B)$ to Avoid_B. Profiles P_A and P_B have $\mathscr{L}(P_A) = A \cup \{\beta\}$ and $\mathscr{L}(P_B) = B \cup \{\beta\}$, where β is a taxon not present in either A or B. The specifics of P_A and P_B are different for strict and loose supertrees.

- For the strict case, we know that for every tree t in P, $A|B$ is either trivial in t or is supported by t. For each tree t in P we do as follows: If $B \cap \mathscr{L}(t) = \emptyset$, put $t|_A$ in P_A, and if $A \cap \mathscr{L}(t) = \emptyset$, put $t|_B$ in P_B. If neither of the previous two cases holds, let t_A be the tree obtained by contracting the minimal subtree of t containing $B \cap \mathscr{L}(t)$ to a single leaf node β and t_B be the tree obtained by contracting the minimal subtree of t containing $A \cap \mathscr{L}(t)$ to a single leaf node β. Put t_A in P_A and t_B in P_B.
- For the loose case, consider again each input tree t in P. By construction, $A|B$ is compatible with t. If $A|B$ is trivial in t or is supported by t, we handle t in the same way as for strict trees. If neither of these conditions holds, then, by compatibility, t must contain a node u with the following property: t can be refined into a tree that supports $A|B$ by splitting u into two nodes u_1 and u_2 joined by an edge, and by making each neighbor of u a neighbor of either u_1 or u_2 as appropriate. Find that u, and then create the edge (u_1, u_2) as just explained. Delete (u_1, u_2), yielding trees

t_1 and t_2 containing u_1 and u_2, respectively. Assume without loss of generality that $A \cap \mathscr{L}(t_2) = \emptyset$ and $B \cap \mathscr{L}(t_1) = \emptyset$. Let t_A be the tree obtained by connecting a leaf node β to u_1 in t_1 and let t_B be the tree obtained by connecting a leaf node β to u_2 in t_2. Add t_A to P_A and t_B to P_B.

In both cases, after the recursive calls of steps 11 and 12, step 13 combines the returned trees T_A and T_B by identifying leaves β and then suppressing the resulting degree-two node.

Verification. The tree T returned by Algorithm 1 is a potential OCT. Even if it is indeed an OCT, we still need to verify its splits to construct a strict or loose supertree. Since the algorithm is heuristic, there is also the possibility that T is not an OCT, in which case we should reject it and look for another tree. We exploit Algorithm 1 to obtain a heuristic that either verifies a potential OCT or rejects it if it is not. The algorithm has one-sided error; that is, it may incorrectly produce a "verified" strict or loose supertree, but if it rejects a tree, then this tree cannot be an OCT.

The heuristic verification algorithm repeatedly picks a random split $A|B$ in T and invokes Algorithm 1 with $A|B$ in the Avoid set. If this yields a tree T' with the same objective value, $A|B$ is eliminated from the supertree. If T' has a lower objective value, the verifier adds $A|B$ to the Avoid set and rejects the tree. If T' has a greater objective value than T, we conclude that $A|B$ is in the supertree and create two subproblems: one for A and one for B (as described earlier in this section). The heuristic verifier then recursively attempts to verify each tree independently, and combines the results from the two calls, as done in Algorithm 1.

As an additional safeguard, we test each potential OCT T by repeatedly rerunning Algorithm 1 using a different starting good splits — perhaps using splits found during the execution of Algorithm 1 itself. If we obtain the same tree T or a different tree with the same objective value, this gives further evidence of the correctness of T. If we get a tree with the same objective value, this information can be used to discard splits from the final supertree, speeding up split verification. If we get a tree with a better score than T, then T is clearly not an OCT. Sufficiently many successful random restarts not only improve our confidence in the quality of OCT, but can also show which edges of an OCT are not in the supertree, accelerating verification.

6 Experimental Results

We developed a prototype implementation of the techniques described in the previous sections, and used it to conduct a series of computational tests. Our system is based on a collection of MATLAB[2] scripts to automatically generate the appropriate ILPs of Section 4.1 directly from the input profiles. These scripts invoke CPLEX[3] to solve the ILPs exactly or to apply the divide-and-conquer heuristics of Section 5.

Our first tests were aimed at finding the limits of the applicability of the exact ILP formulation. As expected, this approach only allowed us to solve rather small problems.

[2] MATLAB is a registered trademark of The MathWorks, Inc.

[3] CPLEX is a trademark of IBM.

The critical parameter is the product of n, the number of taxa, and m, the total number of nontrivial splits in all the input trees. Generally speaking, problems with $nm \geq 2000$ are difficult to solve, while problems with $nm \leq 1000$ are quickly solvable. Thus, we chose $nm = 1000$ as a threshold to use divide and conquer in Algorithm 1. Note that this is just a rule of thumb, as there are problems with $nm > 1000$ that are solvable, such as the smaller of the primates datasets in [16], which has $n = 33$, $m = 48$, and $nm = 1584$.

We analyzed five large data sets using the divide-and-conquer heuristic: the larger of the two primate data sets in [16], seabirds [14], placental mammals [3], legumes [24] and marsupials [6]. The current version of our software assumes a common outgroup among the input trees (i.e., that the trees are rooted). This assumption holds for all but the placental mammal dataset. In the latter, 486 out of the 726 input trees are rooted; the rest are deliberately unrooted. We therefore ran our programs in two different ways on this dataset. In the first, we considered all 726 trees, treating the unrooted trees as rooted. In the second, we took only the 486 rooted trees. Table 1 summarizes the results of our analyses[4].

Table 1. Summary of experimental results. Times are given in seconds or hours. Solution and verification times are listed separately. When the solution was a fan, no verification was needed.

			Strict OCT		Loose OCT	
Data set	n	k	Soln.	Verif.	Soln.	Verif.
Primates	72	24	74 s.	311 s.	240 s.	981 s.
Seabirds	121	7	86 s.	0.59 h.	1.93 h.	3.63 h.
Placental Mammals 1	116	726	608 s.	332 s.	28.4 h.	160 h.
Placental Mammals 2	116	486	282 s.	138 s.	65.8 h.	n.a.
Legumes	571	22	31 s.	fan	n.a	n.a.
Marsupial	267	158	0.97 h.	6.6 h.	4.73 h.	63.6 h.

The supertrees obtained appear quite reasonable[5]. For instance, all 26 of the clusters in the strict marsupial supertree are in the published paper [6]. The loose marsupial supertree contains all clusters in the strict supertree along with 38 additional clusters, most of which are in the published supertree. Out of the 64 clusters in this loose supertree, 61 are in the published paper.

We note that many data sets encountered in practice — in particular, the above-mentioned marsupial and placental mammal data sets — include a "scaffold" tree (also called a "backbone" or "seed" tree); i.e., a tree that covers a broad span of taxa, without necessarily being comprehensive. For instance, the marsupial dataset [6] includes the tree implied by the marsupial classification of [23] "because it is currently widely accepted as a taxonomic reference for mammals, and because its low resolution means it can easily be overruled by more resolved phylogenies, minimizing its influence on the final supertree." However, a low-resolution scaffold has different effects on strict and loose trees. By the definition of strict supertrees, if a set of taxa appears as a fan

[4] Allexperiments were run on an Intel Core 2 64 bit quad-core processor (2.83GHz).

[5] See `http://www.cs.iastate.edu/~fernande/WABIfigures.pdf`.

in any of the input trees, it must remain as a fan in the strict supertree, no matter what the other trees say. Thus, a low-resolution scaffold can lead to a low-resolution strict supertree. On the other hand, again by definition, scaffolds do not prevent loose supertrees from being well-resolved. Our experimental results — e.g., on the marsupial data set — confirm this observation.

An unexpected computational issue encountered during our experiments was the need to deal with *artificial splits*. Informally, these are splits that combine families of taxa in ways that are not justified by the data. More precisely, let $A|B$ be a split that is supported by at least one tree in profile P and that is compatible with all trees in P. Let G be the intersection graph of all sets $C \subseteq B$ such that C is one part of a split (say, $C|D$) in some input tree. Then, $A|B$ is *artificial* if G has more than one connected component.

Artificial splits are not only problematic biologically; they are also a computational nuisance: We can prove that there always exists an OCT without artificial splits, so they never occur in loose or strict supertrees. Unfortunately, they are encountered often, so a lot of time can be spent verifying them, only to discard them. The present version of our system has a mechanism to detect artificial splits. When one is found, we put it in the Avoid set. The presence of artificial splits explains in part the fact that takes longer to find the loose OCT for the second, smaller, placental mammals dataset than the first one. The OCT for the second dataset is not only more resolved, but also many more artificial clusters are found during its construction.

7 Discussion

Loose and strict supertrees provide a rigorous approach for combining phylogenetic information. We have shown that it is possible to construct such supertrees for datasets on the scale of those encountered in practice. From a mathematical standpoint, it would be interesting to elucidate the relationship between our approach and PhySIC's triplet-based approach [16], as well as possible quartet-based versions of the latter. From a practical standpoint, a more detailed performance analysis of loose and strict supertrees is needed to truly evaluate their scalability. To conduct such a study, we first need to improve the efficiency of our prototype implementation. There are a number of ways to do this. Something as simple as using a compiled version of the software should increase the speed notably. Also, at present, the verifier repeats much of the work done by Algorithm 1. Some extra bookkeeping during the execution of the divide-and-conquer algorithm could prevent this duplication of effort. Another direction is to develop constraints to prevent the occurrence of the artificial splits.

References

1. Bansal, M., Burleigh, J.G., Eulenstein, O., Fernández-Baca, D.: Robinson-Foulds supertrees. Algorithms for Molecular Biology 5(1), 18 (2010)
2. Baum, B., Ragan, M.: The MRP method. In: Bininda-Emonds, O. (ed.) Phylogenetic supertrees: Combining Information to Reveal the Tree of Life, pp. 17–34. Kluwer Academic, Dordrecht (2004)
3. Beck, R.M.D., Bininda-Emonds, O.R.P., Cardillo, M., Liu, F.G.R., Purvis, A.: A higher-level MRP supertree of placental mammals. BMC Evol. Biol. 6, 93 (2006)

4. Bininda-Emonds, O.R.P. (ed.): Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life. Series on Computational Biology, vol. 4. Springer, Berlin (2004)
5. Bininda-Emonds, O.R.P., Sanderson, M.J.: Assessment of the accuracy of matrix representation with parsimony analysis supertree construction. Syst. Biol. 50, 565–579 (2001)
6. Cardillo, M., Bininda-Emonds, O.R.P., Boakes, E., Purvis, A.: A species-level phylogenetic supertree of marsupials. J. Zool. 264, 11–31 (2004)
7. Chaudhary, R., Burleigh, J.G., Fernández-Baca, D.: Fast local search for unrooted Robinson-Foulds supertrees. In: Chen, J., Wang, J., Zelikovsky, A. (eds.) ISBRA 2011. LNCS, vol. 6674, pp. 184–196. Springer, Heidelberg (2011)
8. Cotton, J.A., Wilkinson, M.: Majority-rule supertrees. Syst. Biol. 56, 445–452 (2007)
9. Dong, J., Fernández-Baca, D.: Properties of majority-rule supertrees. Systematic Biology 58(3), 360–367 (2009)
10. Dong, J., Fernández-Baca, D., McMorris, F.R.: Constructing majority-rule supertrees. Algorithms in Molecular Biology 5(2) (2010)
11. Dong, J., Fernández-Baca, D., McMorris, F.R., Powers, R.C.: Majority-rule (+) consensus trees. Math. Biosci. 228(1), 10–15 (2010)
12. Goloboff, P.: Minority rule supertrees? MRP, compatibility, and minimum flip display the least frequent groups. Cladistics 21, 282–294 (2005)
13. Gordon, A.D.: Consensus supertrees: The synthesis of rooted trees containing overlapping sets of labelled leaves. J. Classification 9, 335–348 (1986)
14. Kennedy, M., Page, R.D.M.: Seabird supertrees: combining partial estimates of procellariiform phylogeny. The Auk 119(1), 88–108 (2002)
15. McMorris, F.R., Wilkinson, M.: Conservative supertrees. Syst. Biol. 60(2), 232–238 (2011)
16. Ranwez, V., Berry, V., Criscuolo, A., Fabre, P.-H., Guillemot, S., Scornavacca, C., Douzery, E.J.P.: PhySIC: A veto supertree method with desirable properties. Systematic Biology 56(5), 798–817 (2007)
17. Robinson, D.F., Foulds, L.R.: Comparison of phylogenetic trees. Mathematical Biosciences 53, 131–147 (1981)
18. Scornavacca, C.: Supertree methods for phylogenomics. PhD thesis, University of Montpellier II, Montpellier, France (December 2009)
19. Semple, C., Steel, M.: Phylogenetics. Oxford Lecture Series in Mathematics. Oxford University Press, Oxford (2003)
20. Steel, M.A.: The complexity of reconstructing trees from qualitative characters and subtrees. Journal of Classification 9, 91–116 (1992)
21. Swenson, M.S., Suri, R., Linder, C.R., Warnow, T.: An experimental study of quartets maxcut and other supertree methods. Algorithms for Molecular Biology 6, 7 (2011)
22. Wilkinson, M., Thorley, J.L., Pisani, D.E., Lapointe, F.-J., McInerney, J.O.: Some desiderata for liberal supertrees. In: Bininda-Emonds, O.R.P. (ed.) Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life. Series on Computational Biology, vol. 4, pp. 65–85. Springer, Heidelberg (2004)
23. Wilson, D., Reeder, D. (eds.): Mammal Species of the World: A Taxonomic and Geographic Reference, 2nd edn. Smithsonian Institution Press, Washington (1993)
24. Wojciechowski, M., Sanderson, M., Steele, K., Liston, A.: Molecular phylogeny of the Temperate Herbaceous Tribes of Papilionoid legumes: a supertree approach. In: Herendeen, P., Bruneau, A. (eds.) Advances in Legume Systematics, vol. 9, pp. 277–298. Royal Botanic Gardens, Kew (2000)

PepCrawler: A Fast RRT–Like Algorithm for High–Resolution Refinement and Binding–Affinity Estimation of Peptide Inhibitors
(Abstract)

Elad Donsky and Haim J. Wolfson

School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
{eladdons,wolfson}@tau.ac.il

Keywords: RRT, peptide inhibitors, protein-peptide docking, derived peptides, protein-protein interactions, PepCrawler.

Introduction. Design of protein-protein interaction (PPI) inhibitors is a key challenge in Structural Bioinformatics and Computer Aided Drug Design [1, 2]. Peptides, which partially mimic the interface area of one of the interacting proteins, are natural candidates to form protein-peptide complexes competing with the original PPI [3, 4]. Some inhibitory peptides were designed by deriving a short linear segment from one of the proteins in a given PPI complex [5-9]. These peptides were successfully able to inhibit interactions with the partner protein. The prediction of such complexes is especially challenging due to the high flexibility of peptide conformations.

We present PepCrawler, a new tool for deriving binding peptides from protein-protein complexes and prediction of protein-peptide complexes. The implemented algorithm is extremely fast, while allowing backbone flexibility of the peptide combined with side-chain flexibility for both the peptide and the receptor protein. The preliminary results are very promising and in excellent agreement with existing experimental data [8, 9].

Methods. The algorithm accepts as input a structure of a protein-protein complex. First, it derives a relatively short, low-energy binding peptide from one of the interacting proteins. Then, it generates a large amount of *clash-free* peptide docking conformations by using *Rapidly-exploring Random Trees* (RRT) [10, 11] and grid-based collision detection. By computing the binding energy of each of the conformations, an initial protein-peptide conformation is refined to some low-energy docking solutions. Moreover, a dense binding-energy/RMSD plot is created, that assists in evaluating the affinity of the binding solution via energy funnels [12].

Results. We run PepCrawler on 20 input protein-peptide complexes from the PDB, creating the energy plot starting from the native conformation. In 19 complexes we could identify a clear energy funnel, with lowest energy solutions close to the native peptide (up to 1 Å backbone-RMSD). Then, we have tested the PepCrawler refinement conformation mechanism on the previously selected structures. For each structure, we have randomly created few clash-free random peptide conformations, with peptide backbone-RMSD of ~3.5 Å and ~5 Å from the original conformation. In 88% of the 3.5 Å input structures, one of the top-3 PepCrawler refinement solutions had less than 1.6 Å backbone-RMSD from the native conformation. For the 5 Å models, this success rate was 70%.

To test the prediction ability of PepCrawler on "real-life" data, we applied it on 2 cases were in-vivo examined peptide inhibitors were published:

Protein Kinase CK2 Subunit Interaction. An eight-residue peptide was derived from CK2-beta chain (residues 186–193), and shown to be effective inhibitor in-vivo [9]. We have tested PepCrawler on the input CK2 complex. The algorithm derived exactly the same peptide that was used in the in-vivo tests. The refinement solution of this peptide had low binding-energy and its output plot clearly showed a visible steep energy funnel, which indicates its high binding affinity.

HIV-1 Integrase and LEDGF Protein Complex. A short LEDGF-derived peptide was designed containing residues 365-369, which were identified as the main residues that participate in IN binding [8]. In-vivo tests showed that this peptide is a weak IN inhibitor. Although the residues 365-369 are important for IN binding at the protein level, they are not sufficient for inhibition at the peptide level. We run PepCrawler on the IN-LEDGF complex input. The algorithm derived the peptide 365-369 from LEDGF. A significant funnel was not seen in any of the output energy plots. Although few solutions with good complex energies were achieved, the lack of significant energy funnels indicates low binding affinity of this peptide.

Running Time. Compared to other state of the art flexible peptide-protein structure simulations, our algorithm is very fast, and takes only minutes to run on a single PC.

Acknowledgments. This research was supported in part by the Israel Science Foundation [grant no. 1403/09] and the Hermann Minkowski Minerva Geometry Center.

References

1. Arkin, M.R., Whitty, A.: The road less traveled: modulating signal transduction enzymes by inhibiting their protein-protein interactions. Curr. Opin. Chem. Biol. 13, 284–290 (2009)
2. Arkin, M.R., Wells, J.A.: Small-molecule inhibitors of protein–protein interactions: progressing towards the dream. Nature Rev. Drug Discov. 3, 301–317 (2004)
3. Nieddu, E., Pasa, S.: Interfering with Protein-Protein Contact: Molecular Interaction Maps and Peptide Modulators. Curr. Top. Med. Chem. 7, 21–32 (2007)
4. Parthasarathi, L., Casey, F., Stein, A., Aloy, P., Shields, D.C.: Approved drug mimics of short peptide ligands from protein interaction motifs. J. Chem. Inf. Model. 48, 1943–1948 (2008)
5. Hashemzadeh, M., Furukawa, M., Goldsberry, S., Movahed, M.R.: Chemical structures and mode of action of intravenous glycoprotein IIb/IIIa receptor blockers: a review. Exp. Clin. Cardiol. 13, 192–197 (2008)

6. Barr, R.K., Kendrick, T.S., Bogoyevitch, M.A.: Identification of the critical features of a small peptide inhibitor of JNK activity. J. Biol. Chem. 277, 10987–10997 (2002)
7. Phan, J., Li, Z., Kasprzak, A., Li, B., Sebti, S., Guida, W., Schonbrunn, E., Chen, J.: Structure-based Design of High Affinity Peptides Inhibiting the Interaction of p53 with MDM2 and MDMX. J. Biol. Chem. 285, 2174–2183 (2010)
8. Hayouka, Z., Levin, A., Maes, M., Hadas, E., Shalev, D.E., Volsky, D.J., Loyter, A., Friedler, A.: Mechanism of action of the HIV-1 integrase inhibitory peptide LEDGF 361–370. Biochem. Biophys. Res. Commun. 394, 260–265 (2010)
9. Laudet, B., Barette, C., Dulery, V., Renaudet, O., Dumy, P., Metz, A., Prudent, R., Deshiere, A., Dideberg, O., Filhol, O., Cochet, C.: Structure-based design of small peptide inhibitors of pro-tein kinase CK2 subunit interaction. Biochem. J. 408, 363–373 (2007)
10. LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: progress and prospects. Algorithmic and Computational Robotics: New Directions. Peters, 293–308 (2001)
11. Cortes, J., Simeon, T., Ruiz de Angulo, V., Guieysse, D., Remaud-Simeon, M., Tran, V.: A Path Planning Approach for Computing Large-Amplitude Motions of Flexible Molecules. Bioinformatics 21, 116–125 (2005)
12. Zhang, C., Chen, L., DeLisi, C.: Protein-Protein Recognition: Exploring the Energy Funnels Near the Binding Sites. Proteins 34, 255–267 (1999)

Removing Noise from Gene Trees

Andrea Doroftei[1] and Nadia El-Mabrouk[2]

[1] DIRO, Université de Montréal, H3C 3J7, Canada
andreea.doroftei@umontreal.ca
[2] DIRO
mabrouk@iro.umontreal.ca

Abstract. Reconciliation is the commonly used method for inferring the evolutionary scenario for a gene family. It consists in "embedding" an inferred gene tree into a known species tree, revealing the evolution of the gene family by duplications and losses. The main complaint about reconciliation is that the inferred evolutionary scenario is strongly dependant on the considered gene tree, as few misplaced leaves may lead to a completely different history, with significantly more duplications and losses. As using different phylogenetic methods with different parameters may lead to different gene trees, it is essential to have criteria to choose, among those, the appropriate one for reconciliation. In this paper, following the conclusion of a previous paper, we flag certain duplication vertices of a gene tree, the "non-apparent duplication" (NAD) vertices, as resulting from the misplacement of leaves, and consider the optimization problem of removing the minimum number of leaves leading to a tree without any NAD vertex. We develop a polynomial-time algorithm that is exact for two special classes of gene trees, and show a good performance on simulated data sets in the general case.

1 Introduction

Almost all genomes which have been studied contain genes that are present in two or more copies. As an example, duplicated genes account for about 15% of the proteins genes in the human genome [19]. In operational practise, homologous gene copies, e.g. copies in one genome or amongst different genomes that are descended from the same ancestral gene, are identified through sequence similarity. For example, using a BLAST-like method, all gene copies with a similarity score above a certain threshold would be grouped into the same *gene family*. Using a classical phylogenetic method, a *gene tree*, representing the evolution of the gene family by local mutations, can then be constructed based on the similarity scores.

From a functional point of view, grouping genes by sequence similarity is not sufficient to infer a common function for genes. Indeed, it is important to distinguish between two kinds of homologs: *orthologs* which are copies in different species related through speciation, and thus likely to have similar functions, and *paralogs*, which are copies that have evolved by duplication, and more likely to have acquired new functions. Duplication is, indeed, a major source of gene innovation and creation of new functions [21]. In addition, gene losses, arising through the pseudogenization of previously functional genes, also play a key role in the evolution of gene families [2,8,11,17,21]. Understanding the evolution of gene families through speciation, duplication and loss is thus a fundamental question in functional genomics, evolutionary biology and phylogenomics [25,28].

The most commonly used methods to infer evolutionary scenarios for gene families are based on the *reconciliation* approach that compares the species tree S (describing the relationships among taxa) to the gene tree T. Assuming no sequencing errors and a "correct" gene tree, the incongruence between the two trees can be seen as a footprint of the evolution of the gene family through processes other than speciation, such as duplication and loss. The concept of reconciling a gene tree to a species tree under the duplication-loss model was pioneered by Goodman [13] and then widely accepted, utilized and also generalized to models of other processes such as horizontal gene transfer [18,9,27]. Several definitions of reconciliation exist in the literature, one of them expressed in term of "tree extension" [5]. More precisely, a *reconciliation* R between T and S is an extension of T (obtained by grafting new subtrees onto existing branches of T) *consistent* with the species tree, i.e. reflecting the same phylogeny. A duplication and loss history for the gene family is then directly deduced from R. As many reconciliations exist, a natural approach is to select the one that optimizes a given criterion. Natural combinatorial criteria are the number of duplications (duplication cost), losses (loss cost) or both combined (mutation cost) [6,20]. The so called Lowest Common Ancestor (LCA) mapping between a gene tree and a species tree, formulated in [15,24] and widely used [3,10,14,20,22,23,24], defines a reconciliation that minimizes both the duplication and mutation costs.

The main complaint about reconciliation methods is that the inferred duplication and loss history for a gene family is strongly dependant on the gene tree considered for this family. Indeed, a few misplaced leaves in the gene tree can lead to a completely different history, possibly with significantly more duplications and losses [16]. Reconciliation can therefore inspire confidence only in the case of a well-supported gene tree. Typically bootstrapping values are used as a measure of confidence in each edge of a phylogeny. How should the weak edges of a gene tree be handled? A strategy adopted in [6] is to explore the space of gene trees obtained from the original gene tree T by performing Nearest Neighbour Interchanges (NNI's) around weakly-supported edges. The problem is then to select, from this space, the tree giving rise to the minimum reconciliation cost.

In this paper, we explore a different strategy for correcting, or choosing an appropriate gene tree among a set of possible trees, that consists in identifying a number of "misplaced" gene copies in a given gene tree. Criteria for identifying potentially misplaced leaves were given in a previous paper [5], where "non-apparent duplication vertices", were flagged as potentially resulting from the misplacement of leaves in the gene tree. The reason is that each one of these vertices reflects a phylogenetic contradiction with the species tree that is not due to the presence of duplicated gene copies. We develop algorithmic methods for removing the minimum number of leaves resulting in a gene tree T without any non-apparent duplication vertex.

In the next section, we begin by formally introducing our concepts. We then motivate and state our problem in Section 3. Section 4 is dedicated to the algorithmic developments. We first describe two special classes of gene trees which lead to an exact polynomial-time algorithm. We then present a heuristic algorithm for the general case. In Section 5, we test the optimality of our algorithm, and the ability of the presented approach to identify misplaced genes. We finally conclude in Section 6.

2 Definitions

2.1 Trees

In this paper, we only consider rooted trees. Let $\mathcal{G} = \{1, 2, \cdots, g\}$ be a set of integers representing g different species (genomes). A ***species tree*** on $\mathcal{G}$ is a rooted binary tree with exactly g leaves, where each $i \in \mathcal{G}$ is the label of a single leaf (Figure 1(a)). A ***gene tree*** on $\mathcal{G}$ is a rooted binary tree where each leaf is labelled by an integer from $\mathcal{G}$, with possibly repeated leaves (Figure 1(b)). A gene tree represents a gene family, where each leaf labelled i represents a gene copy located on genome i. In the case of a species tree or a *uniquely leaf-labelled gene tree*, i.e. no leaf-label occurs more than once, we will make no difference between a leaf and its label.

Given a tree U, the ***size of*** U, denoted $|U|$, is the number of leaves of U, and the ***genome set of*** U, denoted by $\mathcal{L}(U)$, is the subset of $\mathcal{G}$ defined by the labels of the leaves of U. Given a vertex x of U, U_x is the subtree of U rooted at x, and the ***genome set of*** x, denoted by $\mathcal{L}(x)$, is the subset of $\mathcal{G}$ defined by the labels of the leaves of U_x (for example, in the tree of Figure 1(a), $\mathcal{L}(B) = \{1, 2\}$). If x is not a leaf, we denote by x_l and x_r the two children of x. Finally, if x is not the root, any vertex y on a path from x to the root is an *ancestor* of x.

Given a tree U, a ***leaf removal*** consists in removing a given leaf i from U, and suppress the resulting degree two vertex. A tree U' obtained from U through a sequence of leaf removals is said to be ***included in*** U.

Finally, a subtree U_x of U, for a given vertex x, is said to be a ***maximum subtree*** of U verifying a given property P iff U_x verifies property P and, for any vertex y that is an ancestor of x, U_y does not verify property P.

2.2 Reconciliation

Applying a classical phylogenetic method to the gene sequences of a given gene family leads to a gene tree T that is different from the species tree, mainly due to the presence of multiple gene copies in T, and that may reflect a divergence history different from S. The reconciliation approach consists in "embedding" the gene tree into the species tree, revealing the evolution of the gene family by duplications and losses.

There are several definitions of reconciliation between a gene tree and a species tree [3,10,14,15,20,22,24]. Here we define reconciliation in terms of subtree insertions, following the notation used in [4,14]. We begin by introducing some definitions:

- A *subtree insertion* in a tree T consists in grafting a new subtree onto an existing branch of T.
- A tree T' is said to be an *extension* of T if it can be obtained from T by subtree insertions on the branches of T.
- The gene tree T is said to be *DS-consistent with* S (DS standing for Duplication/Speciation) if T reflects a history with no loss, i.e. if for every vertex t of T such that $|\mathcal{L}(t)| \geq 2$, there exists a vertex s of S such that $\mathcal{L}(t) = \mathcal{L}(s)$ and one of the two following conditions holds:
  (D) either $\mathcal{L}(t_r) = \mathcal{L}(t_\ell)$ (indicating a Duplication),
  (S) or $\mathcal{L}(t_r) = \mathcal{L}(s_r)$ and $\mathcal{L}(t_\ell) = \mathcal{L}(s_\ell)$ (indicating a Speciation).

Definition 1. *A* **reconciliation** *between a gene tree T and a species tree S on $\mathcal{G}$ is an extension $R(T,S)$ of T that is DS-consistent with S.*

For example, the tree of Figure 1(c) is a reconciliation between the gene tree T of Figure 1(b) and the species tree of Figure 1(a). Such a reconciliation between T and S implies an unambiguous evolution scenario for the gene family, where a vertex of $R(T,S)$ that satisfies property (D) represents a duplication (duplication vertex), a vertex that satisfies property (S) represents a speciation (speciation vertex), and an inserted subtree represents a gene loss. The number of duplication vertices of $R(T,S)$ is called the *duplication cost* of $R(T,S)$.

2.3 LCA Mapping

The LCA mapping between T and S, denoted by M, maps every vertex t of T towards the Lowest Common Ancestor (LCA) of $\mathcal{L}(t)$ in S. A vertex t of T is called a ***duplication vertex*** of T with respect to S if and only if $M(t_\ell) = M(t)$ and/or $M(t_r) = M(t)$ (see Figure 1(b)). We denote by $\mathbf{d(T,S)}$ the number of duplication vertices of T with respect to S.

This mapping induces a reconciliation $M(T,S)$ between T and S, where an internal vertex t of T leads to a duplication vertex in $M(T,S)$ iff t is a duplication vertex of T with respect to S. In other words, the duplication cost of $M(T,S)$ is $d(T,S)$ (see for example [3,20,22] for more details on the construction of a reconciliation based on the LCA mapping). Moreover, $M(T,S)$ is a reconciliation that minimizes all of the duplication, loss and mutation costs [5,14]. In particular, $d(T,S)$ is the minimum duplication cost of any reconciliation between T and S.

2.4 Duplication Vertices and MD-trees

Let T be a gene tree. It is immediate to see that any vertex t of T such that $\mathcal{L}(t_\ell) \cap \mathcal{L}(t_r) \neq \emptyset$ (i.e. the left and right subtrees rooted at t contain a gene copy in the same genome) will be a duplication vertex in any reconciliation between

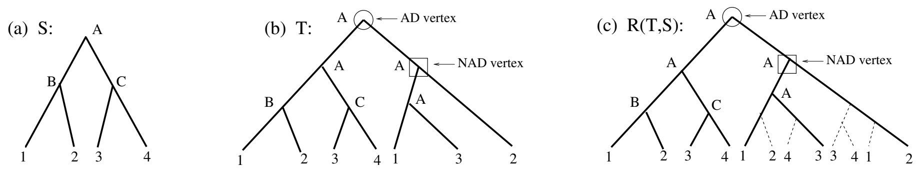


Fig. 1. (a) A species tree S for $\mathcal{G} = \{1, 2, 3, 4\}$. The three internal vertices of S are named A, B and C; (b) A gene tree T. A leaf label x indicates a gene copy in genome x. Internal vertices' labels are attributed according to the LCA mapping between T and S. Flagged vertices are duplication vertices of T with respect to S (Section 2.3); (c) A reconciliation $R(T, S)$ of T and S. Dotted lines represent subtree insertions. The correspondence between vertices of $R(T, S)$ and S is indicated by vertices' labels. Flagged vertices are duplication vertices. All other internal vertices of $R(T, S)$ are speciation vertices. This reconciliation reflects a history of the gene family with two gene duplications preceding the first speciation event, and 4 losses.

T and any species tree S, in particular in $M(T, S)$. Such a vertex is called an ***apparent duplication vertex*** (***AD vertex*** for short) of T. In the tree of Figure 1(b), the root is an AD vertex as its left and right subtree both contain a gene copy in genome 1. Following our notations in [5], given a species tree S, we say that T is a *Minimum-Duplication tree consistent with* S, or equivalently a ***tree that is MD-consistent with*** S, iff the duplication cost $d(T, S)$ is equal to the number of apparent duplications of T. In other words, all duplication vertices of T with respect to S are AD vertices.

However, this is not always true, in other words, a duplication vertex of T with respect to S is not necessarily an AD vertex. We call such a duplication vertex that is not an AD vertex a ***non-apparent duplication vertex***, or simply a ***NAD vertex***. For example, the tree of Figure 1(b) contains one NAD vertex, indicated by a square, and thus T is not MD-consistent with S.

3 Motivation and Problem Statement

Non-apparent duplication vertices point to disagreements between a gene tree and a species tree that are not due to the presence of repeated leaf labels, i.e. multiple copies in the same genome. More precisely, we say that a vertex x of T *splits* three species $\{a, b, c\}$ into $\{a, b; c\}$ if the genome set of one of its children contains a and b but not c, and the genome set of its other child contains c but neither a nor b. Then for any NAD x of T, there is a triplet of species $\{a, b, c\}$ that are split differently by x and by the LCA mapping of x in S. For example, in Figure 1, $\{1, 2, 3\}$ is split into $\{1, 3; 2\}$ by the NAD vertex of T, and into $\{1, 2; 3\}$ by the vertex A in S. It has therefore been suggested that NAD vertices may point at gene copies that are erroneously placed in the gene tree.

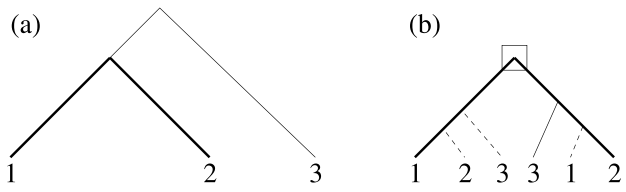


Fig. 2. Let $S = ((1, 2), 3)$ (the tree in (a)) be the phylogenetic tree for the three species $\{1, 2, 3\}$. Let $T = (1, 2)$ be a gene tree. (a), (b) and (c) are the three possibilities for T after a random insertion of a leaf labelled 3. (a) is the only case leading to a tree without any NAD vertex. It reflects a history of the three gene copies without any duplication or loss; (b) and (c) each contains a NAD vertex, and can be explained by a duplication-loss history of minimum mutation cost of 4: 1 duplication and 3 losses.

Different observations made in [5] tend to support this hypothesis. In particular, using simulated data-sets based on the species tree of 12 *Drosophila* species given in [17] and a birth-and-death process, starting from a single ancestral gene, and with different gene gain/loss rates, it has been found that 95% of gene duplications lead to an AD vertex.

Notice however that a misplaced gene in a gene tree T, in other words, a gene randomly placed in T, does not necessarily lead to a NAD vertex. In other words, NAD vertices can only point to a subset of misplaced leaves. However, in the context of reconciliation, the additional damage caused by a misplaced leaf leading to a NAD vertex is the fact that it significantly increases the real mutation-cost of the tree, as shown in Figure 2.

Following the later observations, we exploit the properties of NAD vertices for gene tree correction. If T is not MD-consistent with S, then an MD-consistent tree can always be obtained from T by performing a certain number of leaf removals. Indeed, a gene tree with only two leaves is always MD-consistent with any species tree. The optimization problem considered in this paper is therefore:

Minimum Leaf Removal Problem:

Input: A gene tree T on $\mathcal{G}$ and a species tree S for $\mathcal{G}$;

Output: A tree T^{MAX} included in T and MD-consistent with S of maximum size (i.e. obtained form T by a minimum number of leaf removals).

4 Method

In the rest of this section, we assume that the set of genomes $\mathcal{G}$ and the species tree S for $\mathcal{G}$ are fixed. Let T be a gene tree for a gene family on $\mathcal{G}$. We suppose that T is not an MD-tree consistent with S, i.e. there is at least one duplication vertex of T that is a NAD vertex. We begin by describing special classes of gene trees for which exact polynomial-time algorithms have been developed.

4.1 Uniquely Leaf-Labelled Gene Trees

When the considered gene family contains at most a unique gene copy per genome, the gene tree T is uniquely leaf-labelled. In this case, minimizing the number of leaves that should be removed from T to obtain an MD-tree consistent with S is equivalent to finding the maximum number of genes that lead to the same phylogeny in T and S. In other words, it is immediate to see that the Minimum Leaf Removal Problem reduces, in this case, to the Maximum Agreement Subtree Problem given below.

Maximum Agreement Subtree (MAST) Problem:

Input: A uniquely leaf-labelled gene tree T on $\mathcal{G}$ and a species tree S for $\mathcal{G}$;

Output: A tree T^{MAX} included in T and MD-consistent with S of maximum size.

A more general definition is given in the literature, where the MAST problem is defined on a set of uniquely leaf-labelled trees as the largest tree included in each tree of the set. This definition is equivalent to ours in the case of a gene tree T and a species tree S.

The MAST problem arises naturally in biology and linguistics as a measure of consistency between two evolutionary trees over species or languages, respectively [7]. In the evolutionary study of genomes, different methods and different gene families are used to infer a phylogenetic tree for a set of species, usually yielding different trees. In such a context, one has to find a consensus of the various obtained trees. Considering the MAST problem, introduced by Finden and Gordon [12], is one way to obtain such a consensus. Amir *et al.* [1] showed that computing a MAST of three trees with unbounded degree is NP-hard. However, in the case of two binary trees T and S (which is the case of interest in this paper), the problem is polynomial. The first polynomial-time algorithm for this problem was given by Steel and Warnow [26]. It is a dynamic programming algorithm considering the solution for all pairs of subtrees of T and S; it has a running time of $O(n^2)$, where n is the size of the trees. Later, Cole *et al.* [7] developed an $O(n \log n)$ time algorithm, which, as far as we know, is the most efficient algorithm for solving the MAST problem on two binary trees.

4.2 No AD above NAD

In this section, we consider a tree T containing no AD vertex above a NAD vertex (Figure 3(a)). More precisely, T satisfies Constraint C below:

Constraint C: For each NAD vertex x of T, if y is an ancestor of x that is a duplication vertex, then y is a NAD vertex.

We show, in what follows, that the Minimum Leaf Removal Problem reduces, in this case, to a "generalization" of the Maximum Agreement

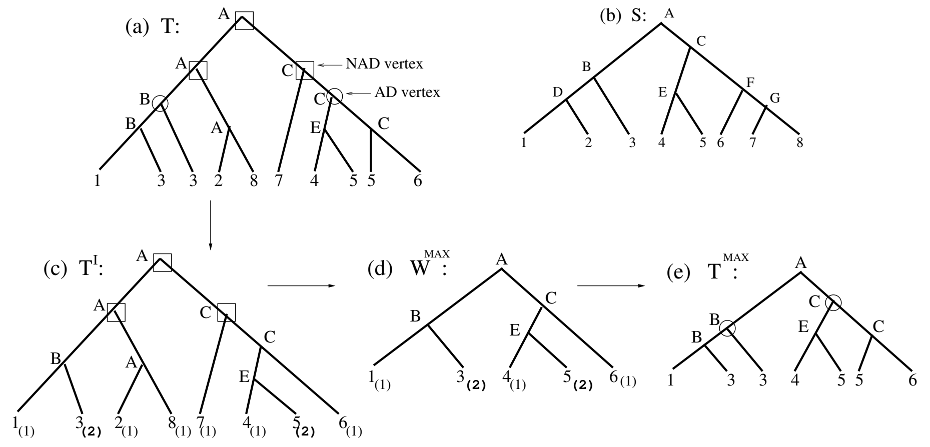


Fig. 3. Solving the Minimum Leaf Removal Problem for a tree satisfying Constraint C; (a) A gene tree T on $\mathcal{G} = \{1, 2, 3, 4, 5, 6, 7, 8\}$; (b) A species tree S for $\mathcal{G}$. Internal vertices of S are identified with different characters. Labels of internal vertices of T are attributed according to the LCA mapping between T and S. T contains 5 duplication vertices with respect to S: two AD vertices (surrounded by a circle) and three NAD vertices (surrounded by a square); (c) The tree T^I obtained by replacing the two subtrees of T rooted at each of the two AD vertices by their weighted induced trees. Leaves' weights are given in brackets; (d) The weighted agreement subtree W^{MAX} of T^I and S of maximum value. $v(W^{MAX}) = 7$; (e) The subtree T^{MAX} of T induced by W^{MAX}. T^{MAX} is an MD-tree consistent with S.

Subtree Problem to weighted trees, where a *weighted tree* is a uniquely leaf-labelled tree with weighted leaves.

Definition 2. *Let U be a tree on $\mathcal{G}$. The* weighted tree U^I induced by (U, S) *is the tree included in S obtained from S by removing all leaves that are not in $\mathcal{L}(U)$, with a weight attributed to each leaf s, representing the number of occurrences of s in U (i.e. the number of leaves of U labelled s).*

Let $T_1, T_2, \cdots T_m$ be the maximum subtrees of T rooted at an AD vertex (i.e. subtrees of T rooted at the highest AD vertices). Then, the tree T^I obtained by replacing each T_i, for $1 \leq i \leq m$, by the weighted tree T_i^I induced by (T_i, S), is a weighted uniquely leaf-labelled tree. An example is given in Figure 3(a), (b) and (c). Let ρ_s be the operation of removing the weighted leaf s from T^I. Then the *corresponding removals* in T consist in removing from T all leaves labelled s.

Finally, we formulate the generalization of the MAST problem to weighted trees as follows, where the value of a weighted tree W is the sum of its leaves' weights.

Weighted Maximum Agreement Subtree (WMAST) Problem:

Input: A weighted tree W on $\mathcal{G}$ and a species tree S for $\mathcal{G}$;

Output: A weighted tree W^{MAX} included in W and MD-consistent with S of maximum value.

We are now ready for the main theorem.

Theorem 1. *Let T be a gene tree satisfying* Constraint C*. Let W^{MAX} be a solution of the WMAST problem on T^I and S, and T^{MAX} be the subtree of T induced by W^{MAX}. Then T^{MAX} is a solution of the* Minimum Leaf Removal Problem *on T and S.*

In other words, Theorem 1 states that solving the Minimum Leaf Removal Problem on T is equivalent to solving the WMAST problem on T^I. We have developed an algorithm (not shown) for solving the WMAST problem, that is a direct generalization of the dynamic programming algorithm of Steel and Warnow [26] to weighted trees, and has the same quadratic running time complexity.

A complete example of the algorithmic methodology used for solving the Minimum Leaf Removal Problem on T and S following Theorem 1 is given in Figure 3. The algorithm will be detailed in the next section.

We now provide a proof of Theorem 1, subdivided into the two following lemmas.

Lemma 1. *The tree T^{MAX} is MD-consistent with S.*

Proof. We show, by contradiction, that T^{MAX} does not contain any NAD vertex. Suppose that T^{MAX} contains a NAD vertex x. Then x maps to the same vertex s of S than one of its child, let say the left child. Then there exist two leaves of $T^{MAX}_{x_l}$, labelled a and b, and one leaf of $T^{MAX}_{x_r}$ labelled c such that the triplet (a, b, c) exhibits a wrong phylogeny. As a non-duplication vertex in T can not become a duplication vertex after leaf removals, we have only two possibilities for x in T:

1. *x is a NAD vertex in T.* Then the genome sets of T_{x_l} and T_{x_r} are disjoint. Moreover, the genome set of $W^{MAX}_{x_l}$ (respec. $W^{MAX}_{x_r}$) is a subset of the genome set of T_{x_l} (respec. T_{x_r}). On the other hand, as x is not a duplication vertex in W^{MAX}, one of the three genes a, b and c should be absent in W^{MAX}_x. And thus, $\{a, b, c\}$ can not be a subset of the genome set of T^{MAX}_x: contradiction.

2. *x is an AD vertex in T.* Then the subtree T_x of T rooted at x contains at least two leaves labelled with the same label d (different from a, b and c), one in T_{x_l} and one in T_{x_r}. Moreover the leaf labelled d in S should belong to the subtree of S rooted at s, and thus to the subtree S_i rooted at the left or right child of s. Such subtree S_i contains at least one leaf labelled a or b or c.

On the other hand, let y be the parent of x in T^I. As an optimal solution of the WMAST problem on T^I removes leaves from the subtree T^I_x, such an operation should result in removing the duplication vertex y. In other words, x and y should map to the same vertex s in S. Moreover the result of the leaf removal from T^I_x should result in a different LCA mapping for x and y. Indeed, otherwise removing leaves from the corresponding subtree in T^I does not contribute to eliminate any NAD from T^I. It follows that S should exhibit the phylogeny $((a, b, c), d)$, which is a contradiction with the result of the last paragraph □

Lemma 2. *Let T' be a tree included in T that is MD-consistent with S. Then $|T'| \leq |T^{MAX}|$.*

Proof. We will show that, for any $s \in \mathcal{G}$, if a leaf i labelled s is removed from T (i.e. i is not a leaf in T'), then all leaves of T labelled s are removed from T.

Suppose this is not the case. Let y be the vertex of T representing the least common ancestor of all leaves labelled s in T. Then y is an AD node. As a leaf i labelled s is removed from T, such removal should contribute in resolving a NAD vertex x of T. From Constraint C, such vertex should be outside the subtree of T rooted at y. Moreover, it should clearly be an ancestor of y (otherwise removing i will have no effect on x).

As x is a NAD vertex, it maps to the same vertex s of S as one of its children, say the left child. Then, there exist two leaves of T_{x_l} labelled a and b, and one leaf of T_{x_r} labelled c such that the triplet (a, b, c) exhibits a wrong phylogeny. Moreover, as removing leaf i labelled s contributes in solving x, we can assume that $a = s$. However, from our assumption, it remains, in T', a leaf labelled s. Thus: (1) either it remains, in T', at least one leaf labelled b and one leaf labelled c, or (2) all leaves labelled b, or all leaves labelled c are removed. In case (1), the wrong phylogeny exhibited by the triplet (a, b, c) is still present, preventing vertex x from being a non-duplication vertex. In case (2), as all copies of b (or equivalently c) are removed, there is no need of removing leaf i labelled s for correcting the wrong phylogeny exhibited by the triple (a, b, c).

Therefore, the weighted tree W' induced by T' is obtained from T^I through a sequence of leaf removals. Now, as W^{MAX} is the solution of the WMAST problem on T^I, then $v(W^{MAX}) \geq v(W')$, and thus $|T^{MAX}| \geq |T'|$ □

4.3 An Algorithm for the General Case

In this section, we present a general algorithm, that is exact in the case of a uniquely leaf-labelled gene tree (Section 4.1) or a gene tree satisfying Constraint C (Section 4.2), and a heuristic in the general case. We first introduce preliminary definitions. For a given tree U (weighted or not), consider the two following properties on U:

Property ONLY-NAD: U has no AD vertices;
Property ONLY-AD: U is rooted at an AD vertex and contains no NAD vertex.

We define the ***NAD-border*** of U as the set of roots of the maximum subtrees of U verifying Property ONLY-NAD, and the ***AD-border*** of U as the set of roots of the maximum subtrees of U verifying Property ONLY-AD.

Algorithm Correct-Tree (Figure 4) is a recursive algorithm that takes as input a gene tree T and a species tree S, and outputs a number of leaf removals transforming T into a tree that is MD-consistent with S. It proceeds as follows:

```
ALGORITHM CORRECT-TREE ($T$, $S$)
1.  LeafRemoval=0;
2.  IF $T$ is a tree MD-consistent with $S$ THEN
3.      RETURN(LeafRemoval)
4.  END IF
5.  $T^I = T$;
6.  FOR ALL $x \in$ AD-border($T$) DO
7.      Replace $T^I_x$ by its induced weighted tree;
8.  END FOR
9.  FOR ALL $x \in$ NAD-border($T^I$) DO
10.     $W_x^{MAX} = WMAST(T^I_x)$;
11.     Replace $T_x$ by the subtree induced by $W_x^{MAX}$;
12.     LeafRemoval = LeafRemoval + $(v(T^I_x) - v(W_x^{MAX}))$;
13. END FOR
14. RETURN(LeafRemoval+CORRECT-TREE($T$, $S$))
```

Fig. 4. An algorithm that takes as input a gene tree T and a species tree S, and outputs the number of leaf removals "LeafRemoval" performed to transform T into a tree that is MD-consistent with S. Here, WMAST points out to an algorithm for solving the WMAST problem.

• Stop condition - Lines 2 to 4: If T is MD-consistent with S, then no leaf removal is performed, and the algorithm terminates.

• Recurrence Loop - Lines 6 to 13: Resolve all maximum subtrees of T verifying CONSTRAINT C as described in Section 4.2, that is:

1. Construct the weighted tree T^I (Lines 6-8);
2. For each root x of a maximum subtree T^I_x of T^I satisfying CONSTRAINT C (Line 9), solve the WMAST problem on T^I_x, which leads to the weighted tree W_x^{MAX} (Line 10), compute the induced tree T_x (Line 11) and store the number of performed leaf removals (Line 12).

If T is a uniquely leaf-labelled tree then $T^I = T$, NAD-border(T^I) is reduced to the root of the tree, and thus loop 9- 13 is just executed once. Moreover, as T^I is unweighted (all labels are equal to 1), WMAST is reduced to MAST. The whole algorithm thus reduces to one resolution of the MAST problem.

If T satisfies CONSTRAINT C, then NAD-border(T^I) is also reduced to the root of the whole tree, and thus loop 9- 13 is just executed once. In this case, the methodology is the one following Theorem 1, and illustrated in Example 3.

In the general case, NAD-border(T^I) is not restricted to a single vertex, and loop 9- 13 can be executed many times. Moreover, at the end of loop 9- 13, the resulting tree is not guaranteed to be MD-consistent with S, as NAD vertices higher than those in NAD-border(T^I) may exist. ALGORITHM CORRECT-TREE may therefore be applied many times.

Complexity: Let n be the size of T. Loop 2- 4 requires to perform the LCA mapping between T and S, and identify AD and NAD vertices. As the LCA mapping can be computed in linear time [29,5], testing whether a tree T is MD-consistent with S can be tested in time $O(n)$. Clearly, Loop 6-8 can be executed in time $O(n)$. As for Loop 9- 13, it has the time complexity of WMAST. As stated in the later section, the $O(n^2)$ algorithm of Steel and Warnow [26] naturally generalizes to the case of weighted trees, and leads to the same running time complexity $O(n^2)$. Therefore, the complexity for one execution of the recursive ALGORITHM CORRECT-TREE is $O(n^2)$. As in the worst case, the algorithm can be executed n times, the total worst case running time complexity is $\mathbf{O(n^3)}$.

Notice that a more efficient algorithm running in $O(n \log(n))$ time exists for the MAST problem [7]. If, as we conjecture, this algorithm can be generalized to the WMAST problem, then it will lead to a time complexity in $O(n^2 \log n)$ for ALGORITHM CORRECT-TREE. This is a short-term perspective for improvement.

5 Results

We only test the optimality of Algorithm Correct-Tree in the case of a gene tree satisfying Property AD-above-NAD, i.e. containing at least one AD vertex above a NAD vertex. Indeed, the algorithm is guaranteed to give the optimal solution otherwise (i.e. for trees satisfying the constraints of Section 4.1 or Section 4.2). We compared the number *NbObtained* of leaf-removal obtained by Algorithm Correct-Tree with the number *NbOptimal* obtained by an exact naive algorithm that tries all possible leaf-subset removals. More precisely, if the minimum number of leaf-removal output by Algorithm Correct-Tree is r, then, we try all subsets of $r-1, r-2 \cdots r-i$ leaf removals, and stop as soon as a tree that is MD-consistent with S is obtained. As the naive algorithm has clearly an exponential-time complexity, tests are performed on trees of limited size.

We considered a genome set of fixed size 5, and gene trees with 6 to 24 leaves. For each size s (from 6 to 24, with steps of 2), we generated 500 random gene trees of s leaves, and kept only those satisfying Property AD-above-NAD. The left diagram of Figure 5 shows that Algorithm Correct-Tree gives an exact solution for more than 65% of the trees (among all of those satisfying Property AD-above-NAD). Moreover, when *NbOptimal* differs from *NbOptimal*, in mot cases the difference is 1. The right diagram of Figure 5 is obtained by averaging, for each size s, the results obtained for all the gene trees of that size. We can see that the error-rate, computed as $(NbObtained - NbOptimal)/NbObtained$, is independent from the size of the tree, and did not exceed 0.15, based on our simulation settings. After testing other dependency factors (non-shown results), it appears that the error-rate only depends on the number of times the loop 9- 13 of Algorithm Correct-Tree is executed, which is not directly related to the number of NADs or ADs in the tree.

Finally, we tested the ability of the presented approach to identify misplaced genes. To do so, we considered a genome set of fixed size 10, and gene trees of size s varying from 10 to 100 (with a step of 10). For a random species tree S and a

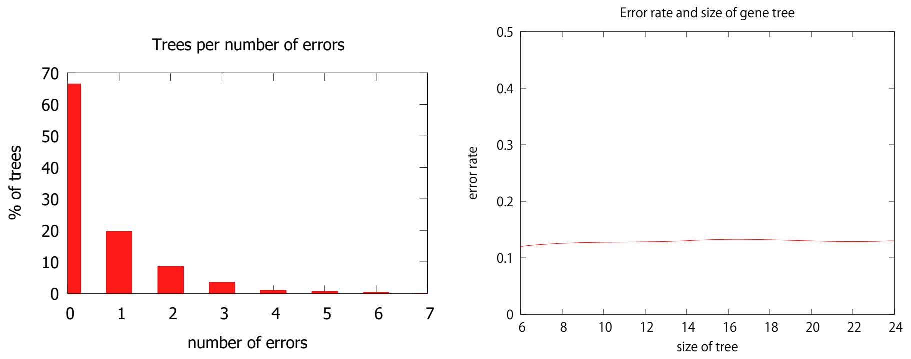


Fig. 5. Comparison of the number *NbObtained* of leaf-removal obtained by Algorithm Correct-Tree with the optimal number *NbOptimal* obtained by an exact algorithm. Left: Percentage of trees leading to a given number *NbObtained* − *NbOptimal* of errors (see text for more details on the used parameters). Right: The error rate, computed as $(NbObtained - NbOptimal)/NbObtained$, depending on the size of the gene tree (number of leaves).

random tree T of size s that is MD-consistent with S, we incorporate randomly $NbAdded = s/10$ leaves with randomly chosen labels. We then test how many "misplaced" leaves our method is able to detect. For each size s, results are averaged over 100 trees. Figure 6 shows the detection percentage of Algorithm

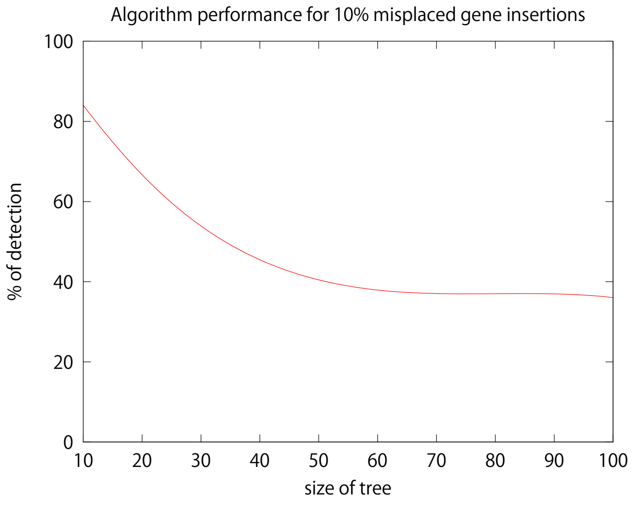


Fig. 6. Percentage of misplaced leaf detection, computed as $(NbObtained/NbAdded) \times 100$, where *NbAdded* is the number of randomly added leaves, and *NbObtained* is the number of leaf removals obtained by Algorithm Correct-Tree (see text for more details)

Correct-Tree, which is computed as (*NbObtained*/*NbAdded*) × 100. This detection percentage decreases with increasing size of the gene tree. This is mainly due to the fact that as an MD-consistent tree needs no leaf removal, its detection percentage is always 100%, and that the more leaves we add (1 for a gene tree of size 10, but 10 for a gene tree of size 100) the less chance we have to end up with an MD-consistent tree. Removing the cases of MD-consistent trees lead to a detection percentage around 40%.

6 Conclusion

Based on observations pointing to NAD vertices of a gene tree as indications of potentially misplaced genes, we developed a polynomial-time algorithm for inferring the minimum number of leaf-removals required to transform a gene tree into an MD-tree, i.e. a tree with no NAD vertices. The algorithm is exact in the case of a uniquely leaf-labelled gene tree, or in the case of a gene tree that does not contain any AD vertex above a NAD vertex. In the general case, our algorithm exhibited results very close to optimality under our simulation settings. Unfortunately, NAD vertices can only reveal a subset of misplaced genes, as a randomly placed gene does not necessarily lead to a NAD vertex. Our experiments show that, on average, we are able to infer 40% of misplaced genes. However, the additional damage caused by a misplaced leaf leading to a NAD is an excessive increase of the real mutation-cost of the tree. Therefore, removing NADs can be seen as a preprocessing of the gene tree preceding a reconciliation approach, in order to obtain a better view of the duplication-loss history of the gene family.

Another use of our method would be to choose, among a set of equally supported gene trees output by a given phylogenetic method, the one that can be transformed to an MD-consistent tree by a minimum number of leaf removals.

A limitation of our approach is that a NAD resulting from a wrong bipartition $\{a, b; c\}$ can be, a priori, solved by removing any gene from this bipartition. Our present approach is able to detect a number of misplaced genes but, in general, it is insufficient to detect precisely the genes that have been erroneously added in the tree. An extension would be to infer all optimal subsets of leaf removals, and to use bootstrapping values on the edges of the tree for a judicious choice of the genes to be removed.

Acknowledgments. We thank Sébastien Langevin for his help in the implementation of the algorithm, and for valuable discussion on the method.

Funding: Research supported by a grant to N.E.M. from the Natural Sciences and Engineering Research Council of Canada.

References

1. Amir, A., Keselman, D.: Maximum agreement subtree in a set of evolutionary trees: matrics and efficient algorithms. SIAM J. Computing 26, 1656–1669 (1997)
2. Blomme, T., Vandepoele, K., De Bodt, S., Silmillion, C., Maere, S., van de Peer, Y.: The gain and loss of genes during 600 millions years of vertebrate evolution. Genome Biology 7, R43 (2006)
3. Bonizzoni, P., Della Vedova, G., Dondi, R.: Reconciling a gene tree to a species tree under the duplication cost model. Theoretical Computer Science 347, 36–53 (2005)
4. Chauve, C., Doyon, J.-P., El-Mabrouk, N.: Gene family evolution by duplication, speciation and loss. J. Comput. Biol. 15, 1043–1062 (2008)
5. Chauve, C., El-Mabrouk, N.: New perspectives on gene family evolution: Losses in reconciliation and a link with supertrees. In: Batzoglou, S. (ed.) RECOMB 2009. LNCS, vol. 5541, pp. 46–58. Springer, Heidelberg (2009)
6. Chen, K., Durand, D., Farach-Colton, M.: Notung: Dating gene duplications using gene family trees. Journal of Computational Biology 7, 429–447 (2000)
7. Cole, R., Farach-Colton, M., Hariharan, R., Przytycka, T., Thorup, M.: An $o(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. SIAM J. Computing 30(5), 1385–1404 (2000)
8. Cotton, J.A., Page, R.D.M.: Rates and patterns of gene duplication and loss in the human genome. Proceedings of the Royal Society of London Series B 272, 277–283 (2005)
9. Doyon, J.-P., Scornavacca, C., Gorbunov, K., Szolloso, G., Ranwez, V., Berry, V.: An effi. algo. for gene/species trees parsim. reconc. with losses, dup. and transf. J. Comp. Biol. 6398, 93–108 (2010)
10. Durand, D., Haldórsson, B.V., Vernot, B.: A hybrid micro-macroevolutionary approach to gene tree reconstruction. Journal of Computational Biology 13, 320–335 (2006)
11. Eichler, E.E., Sankoff, D.: Structural dynamics of eukaryotic chromosome evolution. Science 301, 793–797 (2003)
12. Finden, C.R., Gordon, A.D.: Obtaining common pruned trees. J. Classification 2, 255–276 (1985)
13. Goodman, M., Czelusniak, J., Moore, G.W., Romero-Herrera, A.E., Matsuda, G.: Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. Systematic Zoology 28, 132–163 (1979)
14. Gorecki, P., Tiuryn, J.: DLS-trees: a model of evolutionary scenarios. Theoretical Computer Science 359, 378–399 (2006)
15. Guigó, R., Muchnik, I., Smith, T.F.: Reconstruction of ancient molecular phylogeny. Molecular Phylogenetics and Evolution 6, 189–213 (1996)
16. Hahn, M.W.: Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution. Genome Biology 8(R141) (2007)
17. Hahn, M.W., Han, M.V., Han, S.-G.: Gene family evolution across 12 *drosophilia* genomes. PLoS Genetics 3, e197 (2007)
18. Hallett, M., Lagergren, J., Tofigh, A.: Simultaneous identification of duplications and lateral transfers. In: RECOMB. ACM, New York (2004)
19. Li, W.H., Gu, Z., Wang, H., Nekrutenko, A.: Evolutionary analysis of the human genome. Nature 409, 847–849 (2001)

20. Ma, B., Li, M., Zhang, L.: From gene trees to species trees. SIAM J. on Comput. 30, 729–752 (2000)
21. Ohno, S.: Evolution by gene duplication. Springer, Berlin (1970)
22. Page, R.D.M.: Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. Systematic Biology 43, 58–77 (1994)
23. Page, R.D.M.: Genetree: comparing gene and species phylogenies using reconciled trees. Bioinformatics 14, 819–820 (1998)
24. Page, R.D.M., Charleston, M.A.: Reconciled trees and incongruent gene and species trees. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 37, 57–70 (1997)
25. Sanderson, M.J., McMahon, M.M.: Inferring angiosperm phylogeny from EST data with widespread gene duplication. BMC Evolutionary Biology 7, S3 (2007)
26. Steel, M., Warnow, T.: Kaikoura tree theorems:computing the maximum agreement subtree. Inform. Process. Lett. 48, 77–82 (1993)
27. Tofigh, A., Hallett, M., Lagergren, J.: Simultaneous identification of duplications and lateral gene transfers. IEEE/ACM Trans. Comput. Biol. Bioinform. 8, 517–535 (2011)
28. Wapinski, I., Pfeffer, A., Friedman, N., Regev, A.: Natural history and evolutionary principles of gene duplication in fungi. Nature 449, 54–61 (2007)
29. Zhang, L.X.: On Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. Journal of Computational Biology 4, 177–188 (1997)

Boosting Binding Sites Prediction Using Gene's Positions

Mohamed Elati, Rim Fekih, Rémy Nicolle, Ivan Junier,
Joan Hérisson, and François Képès

ISSB, Genopole - CNRS UPS3201, Université Évry Val d'Essonne,
Genopole Campus 1 - Genavenir 6, 5 rue Henri Desbruères - F-91030 ÉVRY cedex
prenom.nom@issb.genopole.fr

Abstract. Understanding transcriptional regulation requires a reliable identification of the DNA binding sites that are recognized by each transcription factor (TF). Building an accurate bioinformatic model of TF-DNA binding is an essential step to differentiate true binding targets from spurious ones. Conventional approches of binding site prediction are based on the notion of consensus sequences. They are formalized by the so-called position-specific weight matrices (PWM) and rely on the statistical analysis of DNA sequence of known binding sites. To improve these techniques, we propose to use genome organization knowledge about the optimal positioning of co-regulated genes along the whole chromosome. For this purpose, we use learning machine approaches to optimally combine sequence information with positioning information. We present a new learning algorithm called PreCisIon, which relies on a TF binding classifier that optimally combines a set of PWMs and chrommosal position based classifiers. This non-linear binding decision rule drastically reduces the rate of false positives so that PreCisIon consistently outperforms sequence-based methods. This is shown by implementing a cross-validation analysis in two model organisms: *Escherichia coli* and *Bacillus Subtilis*. The analysis is based on the identification of binding sites for 24 TFs; PreCisIon achieved on average an AUC (aera under the curve) of 70% and 60%, a sensitivity of 80% and 70%, and a specificity of 60% and 56% for *B. subtilis* and *E. coli*, respectively.

1 Introduction

Transcription factors (TF) regulate gene expression through the physical interaction with specific cis-regulatory elements termed TF binding sites (TFBS). Genome-wide TFBS identification has drawn substantial interests in the recent years as it represents a critical step in delineating transcription regulatory networks. Previous studies have used both experimental and computational techniques to identify or to predict TFBS. However, traditional experimental techniques, such as DNase I foot-printing and gel-mobility shift assay, are time-consuming and are not suitable for genome-scale studies. While current high-throughput approaches, such as ChIP-chip and CHIP-seq, are more efficient in determining the binding specificity at a large scale [26], they are too costly for daily applications. Efficient computational approaches using cheap and readily available genomic sequence data is therefore most welcome. Such methods can be used, in particular, to complement analysis of high-throughput data. Indeed, binding sites detected by high-throughput *in vitro* methods can be compared with predicted binding sites to prioritize studies aimed at confirming sites that are expected to regulate gene expression *in vivo*.

A number of computational methods have been developed for predicting TFBS given a set of known binding sites. Commonly used methods are based on the definition of a consensus sequence or the construction of a position-specific weight matrix (PWM), where DNA binding sites are represented as a sequence of letter coming from the alphabet $\{A, T, C, G\}$. They then use the PWM to scan new sequences for additional binding sites [25]. Since TFBSs are in general relatively short and possibly degenerate, these approaches systematically lead to a high rate of false positives [26]. Nevertheless, in spite of the wealth of research performed in the area of TFBS prediction, and the many insights gained, achieving a qualitative jump in this field requires information of a conceptually novel type, rather than improvements of methods which all rely essentially on local sequence information [9]. More sophisticated approaches further constrain the set of potential binding sites for a given TF by considering, in addition to PWMs, phylogenetic comparisons [19] or chemical and structural properties [1]. Here we propose to additionally derive useful information from respective gene positioning along the chromosome.

Proper genome-wide coordination of gene expression has been shown to be linked to the spatial organization of genes within the cell [7]. In particular, transcriptional activity is often detected in discrete foci called transcription factories, rather than in a diffuse pattern [5]. These transcription factories gather RNA polymerases, TFs and genes that can be far apart along the DNA [24]. Recent experiments have further shown that genes within a given factory share similar promoter sequences [32] and are co-regulated by the same transcription factors [22]. In this respect, the periodic organization observed for co-regulated genes in *Escherichia coli* [14], for co-expressed [4] and evolutionarily correlated [31] genes in bacteria, has been shown to be crucial for achieving chromosome conformations that favor the formation of such transcription factories. This was demonstrated using a thermodynamic model of chromosome folding where distal transcriptional operators can be cross-linked by TFs [13]. It has been shown, in particular, that chromosomal periodicity favors the formation of solenoid-like structures [15] whereas chromosomal proximity, which is also observed for co-regulated genes [16], favors the formation of rosette-like structures [5]. These two families of chromosomal conformations thus favor the spatial proximity of TF binding sites, which in turn optimizes transcriptional repression or activation [30] through local concentration effects [18]. In other words, there exists a positive feedback loop connecting specific genome organization (periodicity/proximity) to the cellular conformation of chromosomes on to transcriptional control [15].

The rationale proposed in this paper is to combine TFBS nucleic sequence information with gene positioning information to obtain an accurate and robust

TFBS prediction model. This combination must itself be optimized in order to achieve a high classification performance. For this purpose, we model the TF-DNA binding problem as a multi-views classification problem [2]. Combination of multiple classifiers is an important research topic in the field of machine learning, and widely discussed in literature [17]. Methods for classifier fusion range from non-trainable combiners like the majority vote or simple functions, to sophisticated methods that require an additional training step. Other methods such as boosting and bagging [3] have been introduced to cope with the diversity in the classifier opinions. Here, we modify the boosting algorithm AdaBoost [21] to train a TF-DNA binding classifier as an ensemble model, which is a weighted combination of a set of base classifiers trained on different views (gene local sequence and position) of the training data. A key aspect of the boosting technique is that it forces some of the base classifiers to focus on the boundary between positive and negatives examples, thus effectively reducing classification errors. We demonstrate the power of this approach by empirical studies performed on a benchmark dataset from two distinct bacteria: the Gram-negative *Escherichia coli* and the Gram-positive *Bacillus subtilis*.

The paper is organized as follows. We first explain the base classifiers used for TFBS prediction based on the consensus sequence and on the chromosome position regularity of TFBS. We next introduce the classifier fusion algorithm. Results are eventually presented and discussed.

2 PreCisIon: PREdiction of CIS-Regulatory Elements Improved by Gene's Position

PreCisIon is a general method to infer new regulatory relationships between a known TF and all the genes of an organism. It requires two types of data as inputs. First, each gene in the organism must be characterized by some property (view), in our case two views: the promoter sequence and its chromosome position. Second, a list of known regulatory relationships between known TF and some genes is needed. More precisely, for each TF, we need a list of genes known to be regulated by the TF and, if possible, a list of genes known not to be regulated. Such lists can be constructed from publicly available databases of experimentally characterized regulations, *e.g.* RegulonDB for *E. coli* genes (Salgado et al., 2006). While such databases usually do not contain information about the absence of regulations, we discuss below how to generate negative examples. When such data are available, PreCisIon splits the problem of regulatory network inference into many binary classifications from disjoint views. For each view, PreCisIon trains a binary classifier to discriminate between genes known to be regulated and genes known to be not regulated by the TF, based on the data that characterize the genes (*e.g.* classical sequence data). In this paper we introduce a new chromosomal position view to benefit from information pertaining to spatial chromosome conformation. The rationale behind this approach is that, although we make no hypothesis regarding the relationship between the position of a TF and its targets, we assume that if a set of genes is regulated by the same TF, then they are likely to exhibit 1D clustering and/or periodical patterns. The final step is to combine all individual classifiers that are trained in disjoint views. Once trained, the model associated with a given TF is able to assign a class to each new gene which has not been used during training.

2.1 Individual Classifiers

Sequence classifier. Recurrent nucleotide motifs (binding sites) in a collection of DNA sequences (promoter region of target genes) are most commonly modeled by position weight matrices (PWM), which are also called profiles or position-specific scoring matrices. In the general form of this model, an K-long sequence motif is represented by a $4 \times K$ matrix m of weights $w^m_{b,k}$ assigned to each possible base $b[A, C, G, T]$ at each position i in the binding site [25]. A PWM is generally learned from a collection of aligned DNA binding sites that are likely to bound by a common transcription factor (TF). Theoretically, it is formulated as a maximum likelihood problem—finding a PWM such that the likelihood of the observed set of binding sites is maximised. Many alignment driven algorithms have been developed; for instance MotifSampler [27].

Given a learned PWM, the sum of the elements that correspond to a specific sequence s gives a total score for that sequence. This allows the model to provide a binding score BS to all possible binding sites for the protein.

$$BS(s) = \sum_{k=1}^{K} \sum_{b \in [A,C,G,T]} w^m_{b,k} I_{b,k}(s)$$

$I_{b,k}(s) = 1$ if base b occurs at position k of sequence s and 0 otherwise. The higher the score, the more likely a site will be bound by the TF. The Sequence classifier decides that s is a target of the TF whenever $BS(s) > t$ and that it is not a target site, otherwise; t is a threshold decided by some criteria (*e.g.*, ROC curve analysis—see below).

Positional classifier. Recent studies have unveiled regular patterns in the position of cofunctional genes along DNA, both in eukaryotes and bacteria [15,14,31]. Two types of patterns have been identified for co-regulated genes: they tend either to be clustered along the DNA, hereafter referred to as 1D clustering, or to be periodically disposed along the genomes. Both patterns have been shown to facilitate the formation of the so-called transcription factories in the cellular space. More specifically, 1D clustering favors rosettes, while periodicity favors solenoidal arrangements. As a consequence, the position of the genes shall be seen as an expedient classifier for the prediction of transcription factor binding sites. Here, we propose to use both the 1D clustering information and the periodic trends to generate a positional score as an additional classifier. For this purpose, we have recently developed a tool named "Solenoidal Coordinate Method" or "SCM" for analyzing the periodic trends in small sets of positions, which stand here for the positions of target genes [12]. The tool aims i) to detect the presence of a periodic pattern and ii) to allocate to each gene a period-dependent positional score, which reflects the tendency of the gene to be well positioned with

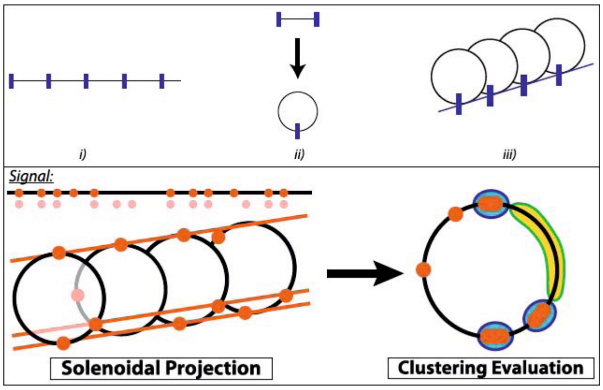


Fig. 1. Principle of SCM, the solenoidal coordinate method. A set of sites (in red, upper left corner) comes from a pattern that periodically repeats at some period P (blurred red points). Some of the initial periodic sites are missing (false negative) or have different positions (noise), and random sites have been added (false positive). Although the resulting pattern seems to be aperiodic, the position of the sites in a solenoidal coordinate of period P (lower left panel) reveals some alignment properties along the solenoidal axis. A projection upon the face view of the solenoid captures exceptionally dense or void regions (rightmost panel).

respect to the other genes at a given period (Fig. 1). By contrast, standard methods based on a Fourier analysis of auto-correlation functions or of pair-distance histograms are poorly adapted to gene position datasets, which are often sparse, noisy and distorted by the presence of false positives and false negatives. Moreover, they provide a collective assessment but do not deliver individual scores, as required here.

SCM is based on a solenoidal representation of the position of the genes. In particular, the presence of a periodic pattern at a certain period leads to an alignment of the genes along the solenoid with the right period or, equivalently, to a clustering of the genes when they are represented on the solenoid face view (Fig. 1). The algorithm is therefore built upon a distance-based information content for the organization of the genes on the solenoid face view, rewarding dense clusters and empty regions (Fig. 1). This information content is computed for all periods, which leads to a spectrum. The peaks that are abnormally high in this spectrum then reveal the periodic tendencies. For both artificial and biological data, the SCM method has been shown to pinpoint periodic regularities that were missed by the standard approaches [12].

At a given period, the SCM allocates a period-dependent positional score to each gene, reflecting its periodic positioning with respect to the other genes. This positional score is equal to the cologarithm of the likelihood for the gene to be periodically positioned with respect to the other genes of the dataset. This likelihood reflects the probability that the density in the vicinity of the gene, on the face view of the solenoid, be obtained with randomized positions.

In other words, the positional score rewards the genes that are located in the dense regions of the solenoid face view (see Fig. 1).

2.2 Classifier Fusion Using Boosting

In this paper we propose a slight variation of the AdaBoost algorithm [21] for the classifier fusion problem. AdaBoost has been shown to improve the prediction accuracy of weak classifiers using an iterative weight update process. The technique combines weak classifiers (classifiers having classification accuracy slightly greater than chance) in a weighted vote, resulting in an overall strong classifier. The proposed variation to the regular form of AdaBoost (Table 1) consists in allowing the algorithm to choose, at each iteration, among weak classifiers trained on different views (here, sequence and position) of the training data. The combination of weights for the final weighting rule is obtained using a shared sampling distribution. In each iteration, a weak classifier is greedily selected from the pool of weak learners trained on disjoint views. This results in a minimization of the training error for the final hypothesis.

Notice that while this is not the regular procedure for training AdaBoost, none of the assumptions on which the algorithm is based is modified; its hypothesis space is only extended by relying on an increased number of classifier types. The weights of the two views of the training example are updated using the same exponential cost function value. Since the same distribution is used for sampling

Table 1. Classifier fusion using shared sampling distribution for boosting

- **Input and initialisation :**
  - N training examples (genes) in a training set T
  - 2 views v (promoter sequence and chromosome position) available for each example and hence 2 training sets such that $T_v = \langle (g_1^v, y_1), (g_2^v, y_2), \cdots, (g_N^v, y_N) \rangle$
  - Given an example g with a class label $c \in (0;1)$, the weight is initialised to $\omega_1(g) = \frac{\sigma_c}{n_c}$ with σ_c the specified sum of class c's weights and n_c the number of genes with the class label c. The sum of all the examples weight must be equal to 1 and respective views for all the training examples are assigned equals resulting in a uniform distribution ω_1.
- **For k=1 to k_{max} do :**
  1. For each view v (promoter Sequence or chromosome Position), train classifiers C_k^v, using the distribution ω_k.
  2. Obtain the errors rates ϵ_k^v of each clasifier : $\epsilon_k^v = P_{i \sim \omega_k}[C_k^v(g_i^v) \neq y_i]$
  3. Select the base classifier C_k^* with the lowest error rate $\epsilon_k^* < 0.5$
  4. Compute the value $\alpha_k^* = \frac{1}{2} ln \frac{1-\epsilon_k^*}{\epsilon_k^*}$
  5. Update examples' weights: For misclassified examples : $\omega_{k+1}(i) = \frac{\omega_i(i) e^{\alpha_k^*}}{Z_k^*}$ and for well classified example: $\omega_{k+1}(i) = \frac{\omega_i(i) e^{-\alpha_k^*}}{Z_k^*}$, where Z_k^* normalises the weights so that ω_{k+1} becomes a normalised distribution.
- **Final hypothesis:** $H(g) = sign(\sum_{k=1}^{kmax} \alpha_k^* C_k^*(g^*)$

training example and updating the weights for all the views, all the convergence proofs that apply to AdaBoost [21] also apply to the proposed version of the algorithm.

3 Results

As a proof of concept, *E. coli* and *B. subtilis* were selected as model organisms. The use of these particularly well-studied model organisms ensures a compete set of annotations. It also allows to cover a broad range of the bacterial phylogeny since *E. coli* is Gram-negative and *B. subtilis* is Gram-positive. In each organism, the Position classifier was built upon the positions of their transcription units (TU). One TU expresses one mRNA but may contain several genes. In this case, it is called an operon and the corresponding genes, the cistrons. The list of operons in *E. coli*, respectively *B. subtilis*, were downloaded from RegulonDB [8], DBTBS [23] respectively. It contains 899 (633 resp.) operons. Genes not present in these operons were further considered, resulting in a total of 3360 (resp. 3426) TUs from the initial 4345 (resp. 4100) genes. For each TU, two features were associated, that is, the promoter sequence and the transcription start position. The tool "retrieve-seq" of "Regulatory Sequence Analysis Tools" (RSAT [28], `http://rsat.ulb.ac.be/rsat/`) was used to retrieve upstream sequences (promoter sequence). Sequence lengths were computed to collect all non-coding sequence up to the first upstream gene, with a maximal distance of 400 bp. Experimentally validated TF-binding sites and gene regulations were downloaded from RegulonDB and DBTBS as well. To avoid having too few positive samples, it was also required that each selected TF should have at least ten positive examples (TUs). Twelve TFs for each model organism satisfy these criteria and were selected.

The Sequence classifier was built using MotifSampler [27], an extension of the Gibbs sampling algorithm for motif finding with a higher-order background model [29]. The background model, usually defined as a n-order Markov model ($n = 0, 1, 2$ or 3), tries to capture all the information in the non-binding sites, which are much more heterogeneous than the binding sites. Since Sequence and Position classifiers each provide an output score for any TU, a threshold is used to discretise the score values to obtain binary decisions (class 0 for non target genes and class 1 for target ones). ROC curves (Fig. 2) can be used to select the optimal decision threshold by maximizing any pre-selected measure of efficiency (*e.g.*, accuracy). Accuracy is not always the best criterion, however, because it is overestimated as a result of the high proportion of negative instances with respect to the positive ones (*e.g.*, 85% accuracy by simply predicting all the cases as negative). In [10], the true rate is proposed to be the summation of the true positive rate and the true negative rate as follows:

$$TrueRate = \frac{TP}{TP + FN} + \frac{TN}{TN + FP}$$

The true rate is more relevant than the standard definition of accuracy when the ratio of postive instances vs negative ones is small or large, which is the case here. In our study we obtain better performance using the True rate. We note that the optimal threshold for the Sequence classifier is around 0.8 for *B. subtilis* and *E. coli*, unlike the Position one which varies from one TF to another.

3.1 Comparison with Individual Classifiers

In order to assess the performance of the proposed method PreCisIon , and compare it with other existing methods, we test it on a benchmark of known regulatory interactions. PreCisIon being a supervised method, a cross-validation procedure is applied to make sure that its performance is measured on cases that have not been used during the model-training step. Consequently the following three-fold cross-validation strategy is adopted: given a TF, a set of genes and operons, and relations between them, the gene set is split randomly in three parts, PreCisIon is trained on two of these subsets and its prediction capacity is measured on the third subset, *i.e.* on those genetic interactions that were not used during training. This process is repeated on all three subset combinations, and the prediction qualities of these three folds are averaged and compared.

The Sequence classifier achieved on average a AUC of 68% and 56%, a sensitivity of 92% and 87%, and a specificity of 30% and 15% for *B. subtilis* and *E. coli*, respectively. Importonly, a weak classifier can be built using only Position information. This classifier achieved a AUC of 51% and 54%, a sensitivity of 25% and 38%, and a specificity of 80% and 78% for *B. subtilis* and *E. coli*, respectively. These results suggest that Position information can be used to improve the specificity of the Sequence classifier. PreCisIon gave rise to the highest performance by the criteria of AUC (70% and 60%, for *B. subtilis* and *E. coli*, respectively), sensitivity (80% and 70%, resp.) and specificity (60% and 56%, resp.). This result is also demonstrated by a ROC analysis (Fig. 2). In general,

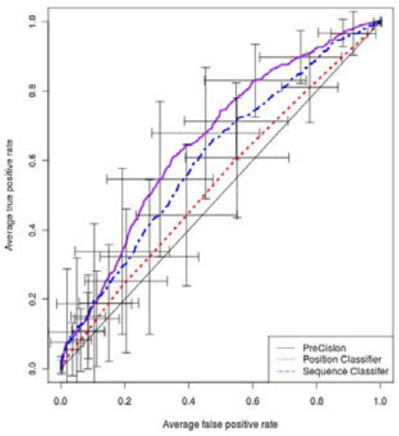


Fig. 2. Receiver Operating Characteristic (ROC) curves on a test set for Position classifier, Sequence classifier and Boosting with classifiers fusion. These ROCs are average curves over all TFs of the ROCs that have been obtained for each single TF.

the ROC curve of an accurate classifier is close to the left-hand and top borders of the plot. PreCisIon is therfore clearly better than the classical Sequence classifier alone. The use of gene positioning information thus improves prediction of TFBS.

3.2 Comparison with Other Classifiers Fusion Methods

In order to find the best combination method for Sequence and Position classifiers, PreCisIon was compared to other approaches ranging from simple non-trainable methods (*e.g.* linear combination) to more sophisticated ones based on an additional training step (*e.g.* stacked generalization, which uses the individual classifiers outputs as input for a second classification step; this allows to weigh the output of individual classifiers). A ROC analysis shows that PreCisIon is more efficient than any other combination methods (Fig. 3).

3.3 Functional Analysis of Predicted Targets of Global TFs

As new TFBSs are predicted purely by computational methods, additional lines of evidence were sought to support the functionality of the new regulatory links obtained for the TFs.

We used the Gene Ontology (GO) enrichment analysis to characterize the biological functions of newly predicted targets of global regulators. We next compared these results against those coming from the set of known targets. For each of the five TFs with the most targets in RegulonDB (CRP, FNR, IHF, ArcA and Fis), we performed the analysis using UniProt GO annotations and the GO enrichment analysis software DAVID [11]. In Table 2, we list for each TF the top ranked GO category among its predicted targets along with the

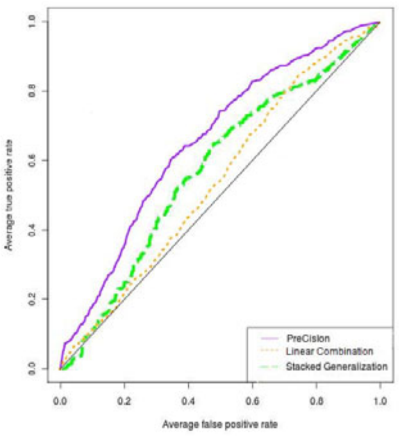


Fig. 3. ROC analysis to compare three classifier fusion algorithms: PreCisIon (classifier fusion using boosting), linear combination based on average and Stacked generalization using Naive Bayes

Table 2. The table shows the most significant GO categories for new predicted gene targets for the *E. coli* TFs using PreCisIon , with the most curated targets in RegulonDB. The table compares the enrichment p-value (p_v) of this category for the newly predicted targets (PT) and known targets (KT).

TF	Top GO Category	p_v PT	p_v KT
CRP	carbohydrate catabolic process	2,7E-9	5.7E-7
FNR	metal ion transport	4.1E-3	8.8E-1
IHF	nitrogen biosynthetic process	2.1E-11	6.0E-9
ArcA	cellular acid biosynthetic process	2.5E-9	1.3E-8
Fis	carbohydrate catabolic process	5,9E-11	5.3E-2

enrichment p-value, as well as the p-value for this category among the curated targets. We observe that for CRP, ArcA and IHF the top ranked GO category based on the predicted targets is significant in the analysis on the curated targets. These results support the assignments made by PreCisIon and indicate that the newly predicted targets for most TFs can be used to correctly extend our understanding of the function of these TFs.

4 Conclusion

How TFs selectively bind to DNA is one of the least understood aspects of TF-mediated regulation of gene expression. An ability to better predict TF binding sites from small training data sets may advance our understanding of TF-DNA binding, and may reveal important insights into TF binding specificity, regulation and coordination of gene expression, and ultimately into gene function. As Sandve and colleagues mentioned [20], another mathematical reformulation of existing approaches will certainly not change the status of the field of TFBS prediction. In this paper we show that for bacteria, chromosomal gene positioning carries significant information. This global positional information fundamentally differs from local sequence information. As a result, they can be seamlessly combined to significantly improve genetic network inference. On this basis, we set up a tool named PreCisIon to optimize this combination using a powerful machine-learning algorithm. Validation on datasets from two phylogenetically remote bacteria shows that PreCisIon improves the specificity of TFBS prediction. On the machine learning side, the work presented here provides a starting point for future investigations of how a strong classifier can be obtained by fusing classifiers from multiple views using boosting techniques.

Future work should focus at i) identifying the influence of transcription factor cooperativity [6] on gene organization and in the definition of chromosomal sectors of functional activity ii) extending the PreCisIon model to others views of transcriptional control, for instance chemical aspects of TF-DNA binding [1].

Acknowledgement. This work was supported by the Sixth European Research Framework (project number 034952, GENNETEC project). We thank anonymous referees for their pertinent suggestions.

References

1. Bauer, A.L., Hlavacek, W.S., Unkefer, P.J., Mu, F.: Using sequence-specific chemical and structural properties of dna to predict transcription factor binding sites. PLoS Comput. Biol. 6 (2010)
2. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the Conf. Computational Learning Theory, pp. 92–100. ACM, New York (1998)
3. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
4. Carpentier, A.S., Torresani, B., Grossmann, A., Henaut, A.: Decoding the nucleoid organisation of *Bacillus subtilis* and *Escherichia coli* through gene expression data. BMC Genomics 6, 84 (2005)
5. Cook, P.R.: Predicting three-dimensional genome structure from transcriptional activity. Nat. Genet. 32 (2002)
6. Elati, M., Neuvial, P., Bolotin-Fukuhara, M., Barillot, E., Radvanyi, F., Rouveirol, C.: Licorn: learning cooperative regulation networks from gene expression data. Bioinformatics 23, 2407–2414 (2007)
7. Fraser, P., Bickmore, W.: Nuclear organization of the genome and the potential for gene regulation. Nature 447, 413–417 (2007)
8. Gama-Castro, S.: Regulondb (version 6.0): gene regulation model of escherichia coli k-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. Nucleic Acids Res. 36, D120–D124 (2008)
9. van Hijum, S.A.F.T., Medema, M.H., Kuipers, O.P.: Mechanisms and Evolution of Control Logic in Prokaryotic Transcriptional Regulation. Microbiol. Mol. Biol. Rev. 73, 481–509 (2009)
10. Hong, C.S.: Optimal threshold from roc and cap curves. Communications in Statistics 38, 2060–2072 (2009)
11. Huang, D.W., Sherman, B.T., Lempicki, R.A.: Systematic and integrative analysis of large gene lists using david bioinformatics resources. Nat. Protocols 4, 44–57 (2008)
12. Junier, I., Herisson, J., Képès, F.: Periodic pattern detection in sparse boolean sequences. Algorithms for Molecular Biology 5, 31 (2010)
13. Junier, I., Martin, O., Képès, F.: Spatial and topological organization of dna chains induced by gene co-localization. PLoS Comput. Biol. 6 (2010)
14. Képès, F.: Periodic transcriptional organization of the e.coli genome. J. Mol. Biol. 340, 957–964 (2004)
15. Képès, F., Vaillant, C.: Transcription-based solenoidal model of chromosomes. ComPlexUs 1, 171–180 (2003)
16. Kolesov, G., Wunderlich, Z., Laikova, O.N., Gelfand, M.S., Mirny, L.A.: How gene order is influenced by the biophysics of transcription regulation. Proc. Natl. Acad. Sci. USA 104, 13948 (2007)
17. Lam, L., Suen, C.Y.: Optimal combinations of pattern classifiers. Pattern Recogn. Lett. 16, 945–954 (1995)
18. Müller-Hill, B.: The function of auxiliary operators. Molecular Microbiology 29, 13–18 (1998)
19. Pennacchio, L., Rubin, E.: Genomic strategies to identify mammalian regulatory sequences. Nat. Rev. Genet. 2, 100–109 (2001)
20. Sandve, G., Drablos, F.: A survey of motif discovery methods in an integrated framework. Biology Direct 1, 11 (2006)

21. Schapire, R.E.: A brief introduction to boosting. In: IJCAI 1999: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pp. 1401–1406. Morgan Kaufmann Publishers Inc., San Francisco (1999)
22. Schoenfelder, S., Sexton, T., Chakalova, L., Cope, N.F., Horton, A., Andrews, S., Kurukuti, S., Mitchell, J.A., Umlauf, D., Dimitrova, D.S., Eskiw, C.H., Luo, Y., Wei, C.L., Ruan, Y., Bieker, J.J., Fraser, P.: Preferential associations between co-regulated genes reveal a transcriptional interactome in erythroid cells. Nature Genetics, 53–61 (2010)
23. Sierro, N., Makita, Y., de Hoon, M., Nakai, K.: Dbtbs: a database of transcriptional regulation in bacillus subtilis containing upstream intergenic conservation information. Nucleic Acids Res. 36, 93–96 (2008)
24. Spilianakis, C.G., Lalioti, M.D., Town, T., Lee, G.R., Flavell, R.A.: Interchromosomal associations between alternatively expressed loci. Nature 435, 637–645 (2005)
25. Stormo, G.: Dna binding sites: representation and discovery. Bioinformatics 16, 16–23 (2000)
26. Stormo, G.D.: Determining the specificity of protein-DNA interactions. Nat. Rev. Genet. 11, 751–760 (2010)
27. Thijs, G., Lescot, M., Marchal, K., Rombauts, S., Moor, B.D., Rouzé, P., Moreau, Y.: A higher-order background model improves the detection of promoter regulatory elements by gibbs sampling. Bioinformatics 17, 1113–1122 (2001)
28. Thomas-Chollier, M., Sand, O., Turatsinze, J.V., Janky, R., Defrance, M., Vervisch, E., Brohée, S., van Helden, J.: Rsat: regulatory sequence analysis tools. Nucleic Acids Res. 36, 119–127 (2008)
29. Thompson, W., Rouchka, E.C., Lawrence, C.E.: Gibbs recursive sampler: finding transcription factor binding sites. Nucleic Acids Res. 31, 3580–3585 (2003)
30. Vilar, J.M.G., Leibler, S.: DNA looping and physical constraints on transcription regulation. J. Mol. Biol. 331 (2003)
31. Wright, M., Kharchenko, P., Church, G., Segrè, D.: Chromosomal periodicity of evolutionarily conserved gene pairs. PNAS 104 (2007)
32. Xu, M., Cook, P.R.: Similar active genes cluster in specialized transcription factories. J. Cell. Biol. 181, 615–623 (2008)

Constructing Perfect Phylogenies and Proper Triangulations for Three-State Characters

Rob Gysel, Fumei Lam, and Dan Gusfield

Department of Computer Science, University of California, Davis, 1 Shields Avenue, Davis CA 95616, USA
rsgysel@ucdavis.edu, {flam,gusfield}@cs.ucdavis.edu

Abstract. In this paper, we study the problem of constructing perfect phylogenies for three-state characters. Our work builds on two recent results. The first result states that for three-state characters, the local condition of examining all subsets of three characters is sufficient to determine the global property of admitting a perfect phylogeny. The second result applies tools from minimal triangulation theory to the partition intersection graph to determine if a perfect phylogeny exists. Despite the wealth of combinatorial tools and algorithms stemming from the chordal graph and minimal triangulation literature, it is unclear how to use such approaches efficiently to construct a perfect phylogeny for three-state characters when the data admits one. We utilize structural properties of both the partition intersection graph and the original data in order to achieve a competitive time bound.

1 Introduction

In this paper, we study the problem of constructing phylogenies, or evolutionary trees, to describe ancestral relationships between a set of observed taxa. Each taxon is represented by a sequence and the evolutionary tree provides an explanation of branching patterns of mutation events transforming one sequence into another.

We will focus on the widely studied *infinite sites* model from population genetics, in which the mutation of any character can occur at most once in the phylogeny. Without recombination, the phylogeny is a tree called a Perfect Phylogeny. The problem of determining if a set of binary sequences fits the infinite sites model without recombination corresponds to determining if the data can be derived on a perfect phylogeny. A generalization of the infinite sites model is the *infinite alleles* model, in which any character can mutate many times but each mutation of the character must lead to a distinct allele (state). Again, without recombination, the phylogeny is tree, called a *multi-state* Perfect Phylogeny. Correspondingly, the problem of determining if multi-state data fits the infinite-alleles model without recombination corresponds to determining if the data can be derived on a multi-state perfect phylogeny.

Dress and Steel [7] and Kannan and Warnow [12] both give algorithms that construct perfect phylogenies for three-state characters when one exists.

The goal of this work is to extend the results in [15] using the minimal separators of the partition intersection graph to create a three state construction algorithm that is competitive with Dress and Steel's algorithm.

2 Notation and Prior Results

The input to our problem is a set of n sequences (representing taxa), where each sequence is a string of length m over r states. The input can be considered as a matrix of size $n \times m$, where each row corresponds to a sequence and each column corresponds to a character (or site). We denote characters by $\mathcal{C} = \{\chi^1, \chi^2, \chi^3, \ldots, \chi^m\}$ and the states of character χ^i by χ^i_j for $0 \leq j \leq r-1$. A *species* s is any sequence $s_1, s_2, \ldots, s_m \in \{\chi^1_{j_1}, \chi^2_{j_2}, \ldots \chi^m_{j_m} : 0 \leq j \leq r-1\}$, where s_i is the *state* of character χ^i for s. Note that a species need not be one of the input sequences.

The *perfect phylogeny problem* is to determine whether an input set S can be displayed on a tree such that

1. each sequence in input set S labels exactly one node in T,
2. each leaf is labeled by a sequence in S,
3. each vertex of T is labeled by a species,
4. for every character χ^i and for every state χ^i_j of character χ^i, the set of all vertices in T labeled by species whose state of character χ^i is χ^i_j forms a connected subtree of T.

The general perfect phylogeny problem (with no constraints on r, n, and m) is NP-complete [4,19]. However, the perfect phylogeny problem becomes polynomially solvable (in n and m) when r is fixed. For $r = 2$, this follows from the Splits Equivalence Theorem [9,18]. For $r = 3$, Dress and Steel gave an $O(nm^2)$ algorithm [7] and for $r = 3$ or 4, Kannan and Warnow gave an $O(n^2m)$ algorithm [12]. For any fixed r, Agarwala and Fernández-Baca gave an $O(2^{3r}(nm^3 + m^4))$ algorithm [1], which was improved to $O(2^{2r}nm^2)$ by Kannan and Warnow [13]).

Definition 1 ([5,18]). *For a set of input sequences M, the* partition intersection graph *$G(M)$ is obtained by associating a vertex for each character state and an edge between two vertices χ^i_j and χ^k_l if there exists a sequence s with state j in character $\chi^i \in \mathcal{C}$ and state l in character $\chi^k \in \mathcal{C}$. For a subset of characters $\Phi = \{\chi^{i_1}, \chi^{i_2}, \ldots, \chi^{i_k}\}$, let $G(\Phi)$ denote the partition intersection graph $G(M)$ restricted to the characters in Φ.*

Note that by definition, there are no edges in the partition intersection graph between states of the same character.

Definition 2. *A graph H is* chordal*, or* triangulated*, if there are no induced chordless cycles of length four or greater in H.*

See [6] and [3] for further details on chordal graphs.

Consider coloring the vertices of the partition intersection graph $G(M)$ as follows. For each character χ^i, assign a single color to the vertices $\chi_0^i, \chi_1^i, \ldots, \chi_{r-1}^i$. A *proper triangulation* of the partition intersection graph $G(M)$ is a chordal supergraph of $G(M)$ such that every edge has endpoints with different colors. In [5], Buneman established the following fundamental connection between the perfect phylogeny problem and triangulations of the corresponding partition intersection graph.

Theorem 1. *[5,18] A set of taxa M admits a perfect phylogeny if and only if the corresponding partition intersection graph $G(M)$ has a proper triangulation.*

In particular, pairs of character states in the same node of a perfect phylogeny not appearing in a shared row of M specifies the edges to add to properly triangulate $G(M)$. Conversely, if M has a perfect phylogeny, it can be constructed from the *clique tree* of a proper triangulation of $G(M)$. T is a clique tree for a graph G if

1. the nodes of T are in bijection with the maximal cliques of G,
2. for each vertex v of G, the maximal cliques containing v form a connected subtree of T.

In recent work, it is shown that there is a complete description of minimal obstruction sets for three-state characters analogous to a well-known result on obstruction sets for binary characters (the four gamete condition) [15]. These results allow us to expand upon recent work of Gusfield which uses properties of triangulations and minimal separators of partition intersection graphs to solve several problems related to multi-state perfect phylogenies [10].

An (a, b)-*separator* of graph G is a set of vertices whose removal from G separates a and b. A *minimal* (a, b)-*separator* is an (a, b)-separator such that no proper subset is an (a, b)-separator and a *minimal separator* is a separator that is a minimal (a, b)-separator for some pair of vertices a and b. For a set of vertices X, let $G - X$ be the induced subgraph of G after removing vertices X. If S and T are two minimal separators of G, we say S is *parallel* to T if there is a single connected component C of $G - T$ such that $S \subseteq C \cup T$ (otherwise S and T *cross*). A pair of vertices a and b *cross* S if S is a (a, b)-separator. The *neighborhood* of a set of vertices X is $N(X) = \{v \in G - X : (u, v) \in E(G) \text{ for some } u \in X\}$. A component C of $G - S$ is *full* if the neighborhood $N(C)$ is equal to S.

In a graph whose vertices are assigned colors, a *legal separator* is a separator such that no two vertices have the same color. Let Δ_G denote the minimal separators of graph G. For vertices X in G, we *saturate* X by adding all edges between vertices in X to create a clique. For $\mathcal{X}$ a collection of vertex subsets, $G_{\mathcal{X}}$ denotes the graph obtained by saturating every $X \in \mathcal{X}$.

A proper triangulation of $G(M)$ is *minimal* if no proper subgraph is a proper triangulation of $G(M)$. Parra and Scheffler [16,17] characterized minimal triangulations of arbitrary graphs by extending results from Kloks, Kratsch, and Spinard [14].

Theorem 2 (Minimal Triangulation Theorem [16,17]). *Suppose $Q \subseteq \Delta_G$ is a maximal set of pairwise parallel minimal separators of G. Then G_Q is a* *minimal triangulation of G and $\Delta_{G_Q} = Q$. Conversely, if H is a minimal triangulation of G, then Δ_H is a maximal pairwise parallel set of minimal separators of G.*

The following are necessary and sufficient conditions for the existence of a perfect phylogeny for data over arbitrary number of states. We refer the reader to [10] for the proofs.

Theorem 3 (Theorem 2 (MSP) [10]). *For input M over r states ($r \geq 2$), there is a perfect phylogeny for M if and only there is a set Q of pairwise parallel legal minimal separators in $G(M)$ such that every illegal minimal separator in $G(M)$ is crossed by at least one separator in Q.*

Theorem 4 (Theorem 3 (MSPN) [10]). *For input M over r states ($r \geq 2$), there is a perfect phylogeny for M if and only if there is a set Q of pairwise parallel legal minimal separators in partition intersection graph $G(M)$ such that every mono-chromatic pair of nodes in $G(M)$ is separated by some separator in Q.*

For the special case of input M with characters over three states ($r = 3$), the partition intersection graph satisfies additional structure and the following theorems give necessary and sufficient conditions for the existence of a perfect phylogeny for M [15].

Theorem 5. *([15]) Given an input set M with at most three states per character ($r \leq 3$), M admits a perfect phylogeny if and only if every subset of three characters of M admits a perfect phylogeny.*

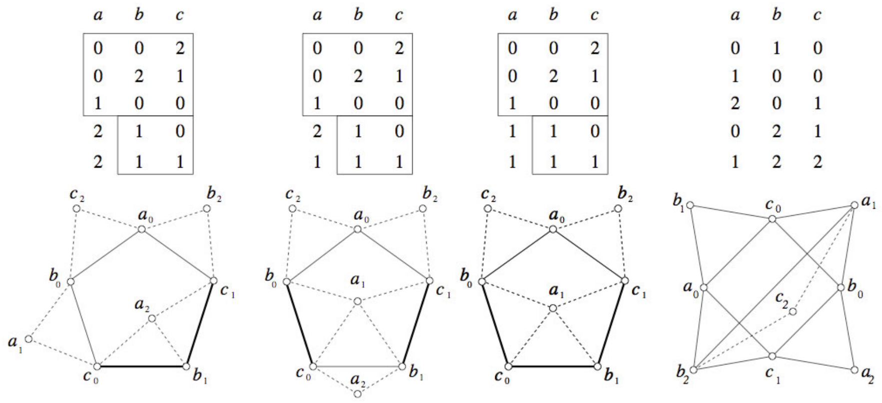
Furthermore, there is an explicit description of all minimal obstruction sets to the existence of a perfect phylogeny.

Theorem 6. *([15]) For input M over 3-state characters, there exists a perfect phylogeny for M if and only if both of the following conditions hold:*

1. *for every pair of columns of M, the partition intersection graph induced by the columns is acyclic,*
2. *for every triple of columns of M, the partition intersection graphs induced by the columns does not contain any of the graphs shown in Figure 1 up to relabeling of the character states.*

This complete characterization of minimal obstruction sets allows us to simplify Theorem 4 in the case $r = 3$.

Theorem 7. *([15]) For input M on at most three states per character ($r \leq 3$), there is a three-state perfect phylogeny for M if and only if the partition intersection graph for every pair of characters is acyclic and every monochromatic pair of vertices in $G(M)$ is separated by a legal minimal separator.*



a	b	c
0	0	2
0	2	1
1	0	0
2	1	0
2	1	1

a	b	c
0	0	2
0	2	1
1	0	0
2	1	0
1	1	1

a	b	c
0	0	2
0	2	1
1	0	0
1	1	0
1	1	1

a	b	c
0	1	0
1	0	0
2	0	1
0	2	1
1	2	2

c_2 b_2 a_0 b_0 c_1 a_2 a_1 c_0 b_1
c_2 b_2 a_0 b_0 a_1 c_1 c_0 a_2 b_1
c_2 b_2 a_0 b_0 a_1 c_1 c_0 b_1
b_1 c_0 a_1 a_0 b_0 c_2 b_2 c_1 a_2

Fig. 1. Minimal obstruction sets for three-state characters up to relabeling. The boxes highlight the input entries that are identical for three of the obstruction sets.

Theorem 7 shows that the requirement of Theorem MSPN that the legal minimal separators in Q be pairwise parallel, can be removed for the case of input data over three-state characters. The condition in Theorem 7 that the input is over three state characters is necessary, as there are examples showing that the theorem does not extend to data with four state characters.

It is known that all of the legal minimal separators for three-state input can be found in $O(nm^2)$ time and the algorithm to check if each mono-chromatic pair is separated by a legal minimal separator can be performed during the algorithm for generating the legal minimal separators (see below). Therefore, the 3-state perfect phylogeny *decision* problem can be solved in $O(nm^2)$ time using minimal separators. However, it is not clear how minimal separators can be used to solve the *construction* problem in a similar time bound. In [10], Gusfield used the minimal separator approach and integer linear programming methods to solve both the decision and construction problem for k-state perfect phylogeny. Since integer linear programming methods in general do not have polynomial time bounds, this naturally leads to the following question: is there an $O(nm^2)$ algorithm for the construction problem for 3-state perfect phylogeny using the separator approach?

In the next section, we study the structure of separators in the partition intersection graph for 3-state input. We first state two lemmas from [15].

Lemma 1. *(Lemma 3.4 [15]) Let M be a set of input species on three characters a, b, and c with at most three states per character. If partition intersection graph $G(a, b, c)$ is properly triangulatable, then the only possible chordless cycles in $G(a, b, c)$ are chordless 4-cycles, with two colors appearing once and the remaining color appearing twice.*

Lemma 1 implies that if a subset of three characters χ^i, χ^j, χ^k in M is properly triangulatable, then there is a unique set of edges $F(\chi^i, \chi^j, \chi^k)$ that must be added to triangulate the chordless cycles in $G(\chi^i, \chi^j, \chi^k)$. Construct a new graph $G'(M)$ on the same vertices as $G(M)$ with edge set $E(G(M)) \cup \{\cup_{1 \leq i<j<k \leq m} F(\chi^i, \chi^j, \chi^k)\}$. $G'(M)$ is the partition intersection graph $G(M)$ together with additional edges to properly triangulate all chordless cycles in $G(\chi^i, \chi^j, \chi^k)$ $(1 \leq i < j < k \leq m)$. In $G'(M)$, edges from the partition intersection graph $G(M)$ are called E-edges and edges that have been added as triangulation edges for some triple of columns are called F-edges.

Lemma 2 (Lemmas 4.2, 4.3, 4.7 [15]). *Let M be a set of input species on three characters a, b, and c with at most three states per character. Then $G'(M)$ cannot contain a chordless cycle with one or more F-edges. If C is a chordless cycle in $G'(M)$ with only E-edges, then C has length exactly four with four distinct colors.*

3 Structure of Separators

In this section, our goal is to study the structure of minimal separators in $G(M)$ and $G'(M)$, with the ultimate goal of showing that it suffices to consider only the legal minimal separators of $G(M)$ while disregarding the illegal minimal separators. We will prove that when M has a perfect phylogeny, the minimal separators of $G'(M)$ are exactly the legal minimal separators of $G(M)$. We build on techniques developed in [10] to generate the minimal separators of $G'(M)$ and their parallel relations in $O(nm^2)$ time. This will allow us to use a greedy approach to pick a maximal pairwise parallel set of legal minimal separators. These minimal separators will then define a set of fill edges, and a perfect phylogeny will be constructed in the form of a clique tree using Maximum Cardinality Search (MCS). MCS was introduced by Tarjan and Yannakakis [20] and may be used to recognize chordal graphs. Blair and Peyton [3] showed how MCS can be used to construct a clique tree for a chordal graph in linear time.

We now prove the following theorem on the separator structure of $G'(M)$.

Theorem 8. *Let M be a set of taxa over 3-state characters. M allows a perfect phylogeny if and only if the graph $G'(M)$ (the partition intersection graph $G(M)$ together with F-edges) does not contain any illegal minimal separators.*

Proof. Suppose M allows a perfect phylogeny and suppose there is an illegal minimal separator S in $G'(M)$ with a monochromatic pair of vertices u and v. Since S is a minimal separator, there exist two full components C, D of $G-S$ and paths connecting u and v in $C \cup \{u, v\}$ and $D \cup \{u, v\}$. Consider the shortest such paths P_C and P_D respectively (note that there are no chords within P_C and no chords within P_D). Since C and D are components separated by S, there are no edges between C and D. Also, u and v are not adjacent in $G'(M)$ since u and v have the same color and $G'(M)$ contains no illegal edges. This implies the union of P_C and P_D creates a chordless cycle. By Lemma 2, $G'(M)$ cannot contain any chordless cycles of length five or greater, so the union of the paths P_C and P_D must be a four cycle C and in particular, must be a cycle $u \to x \to v \to x' \to u$, where u and v have the same color. C is a chordless four cycle in $G'(M)$ on at most three colors, which cannot occur since we have triangulated all such cycles by F-edges. This contradiction implies S cannot be an illegal minimal separator.

Now, suppose $G'(M)$ does not contain any illegal minimal separators. By Theorem 2, graph $G'(M)$ has a proper triangulation and since $G(M)$ is a subgraph of $G'(M)$, $G(M)$ also has a proper triangulation. It follows that M has a perfect phylogeny. □

This suggests that analyzing the minimal separators of $G'(M)$ suffices for 3-state construction. In order to use techniques in [10], the goal of our next two results will be to show how $\Delta_{G'(M)}$ can be constructed using $\Delta_{G(M)}$.

Lemma 3. *Let M be a set of 3-state taxa with perfect phylogeny. Then H is a minimal triangulation of $G'(M)$ if and only if H is a minimal proper triangulation of $G(M)$.*

Proof. Suppose H is a minimal proper triangulation of $G(M)$. Each F-edge comes from a chordless cycle of length four on three colors (see Lemma 1), so this edge must appear in any proper triangulation of $G(M)$. Thus the F-edges must be edges of H, so $G'(M) \subseteq H$ and H is a proper triangulation of $G'(M)$. If H is not minimal with respect to $G'(M)$, there is some H' where $G'(M) \subseteq H' \subset H$ and thus $G(M) \subseteq H' \subset H$, contradicting the minimality of H. Thus H is a minimal proper triangulation of $G'(M)$.

Conversely, suppose H is a minimal triangulation of $G'(M)$. H is a proper triangulation of $G(M)$ by Theorem 8, and if there is some $G(M) \subseteq H' \subset H$ the F-edges must be edges of H' or H' has a chordless four cycle. H' is then a proper triangulation of $G'(M)$, a contradiction, so H must be a minimal proper triangulation of $G(M)$. □

Let $\Delta^L_{G(M)}$ denote the set of legal minimal separators of $G(M)$.

Theorem 9. *Suppose M is a set of taxa with 3-state characters that has a perfect phylogeny. Then the legal minimal separators of $G(M)$ are exactly the minimal separators of $G'(M)$.*

Proof. Let S be a minimal separator of $G'(M)$, and suppose Q is a set of maximal pairwise parallel minimal separators of $G'(M)$ with $S \in Q$. $G'(M)_Q$ is a minimal triangulation of $G'(M)$ by Theorem 2, and $G'(M)_Q$ is a minimal proper triangulation of $G(M)$ by Lemma 3. By Theorem 2, $\Delta_{G'(M)_Q} = Q$ so $S \in \Delta_{G'(M)_Q}$, and $\Delta_{G'(M)_Q} \subseteq \Delta_{G(M)}$ so $S \in \Delta_{G(M)}$. S is legal by Theorem 8, so $\Delta_{G'(M)} \subseteq \Delta^L_{G(M)}$.

Conversely, let $S \in \Delta^L_{G(M)}$. Assume that S is not a minimal separator of $G'(M)$. First we show that there must be an F-edge crossing S (i.e. $f = xy$ where x and y cross S). Suppose for the sake of contradiction that no F-edge crosses S. Then every full connected component C of $G(M) - S$ is also a connected component of $G'(M) - S$, and $N_{G(M)}(C) \subseteq N_{G'(M)}(C)$ so $N_{G'(M)}(C) = S$. Thus S has two or more full components and is a minimal separator of $G'(M)$, a contradiction. Therefore some F-edge $f = xy$ exists where x and y cross S. By definition of F-edges, there is a four cycle $x \rightarrow u \rightarrow y \rightarrow v \rightarrow x$ in $G(M)$ with monochromatic pair u, v that witnesses f. S is an $xy-$separator and must contain u and v. Therefore S is illegal, a contradiction, so S is a minimal separator of $G'(M)$. □

Combining Theorems 2, 8, and 9, we obtain the following.

Theorem 10. *Let M be an instance of 3-state perfect phylogeny. Then M has a perfect phylogeny if and only if any maximal pairwise parallel set of legal minimal separators Q of $G(M)$ induces a proper triangulation $G(M)_Q$ of $G(M)$.*

We are now ready to state our algorithm.

Algorithm. Proper Triangulation for 3-State Characters

1. Compute $\Delta^L_{G(M)}$ using proper clusters [1,10,13].
2. Stop if there is a monochromatic pair not separated by any legal minimal separator.
3. Compute the crossing relations for $\Delta^L_{G(M)}$.
4. Greedily construct a maximal pairwise parallel subset Q of $\Delta^L_{G(M)}$.
5. Add edges to $G(M)$ to make each $S \in Q$ a clique. Call this graph G_Q.
6. Use MCS to construct a clique tree for G_Q in $O(|V(G_Q)| + |E(G_Q)|)$ time [3].

We analyze the complexity of our algorithm by showing that each step is $O(nm^2)$. The first step uses concepts from [1,10,13], which we detail here for completeness.

Lemma 4. *[8] Let S be a subset of the vertices of a graph G. Then S is a minimal separator of G if and only if $G - S$ has two or more full components.*

A *proper cluster* is a bipartition of the taxa such that each character shares at most one state across the bipartition [1,13]. There are $O(m)$ proper clusters when k is fixed ($O(2^k)$ per character). Here, we elaborate on how each $S \in \Delta^L_{G(M)}$ is *witnessed* by a proper cluster as discussed in [10]. For a connected component C of $G(M) - S$, let $t(C) = \{t_i \mid t_i$ witnesses a vertex $v \in C\}$. We call $\{t(C) \mid C$ is a connected component of $G(M) - S\}$ the *$S-$partition* of the taxa. $G(M) - S$ has two or more full components C_1 and C_2. Place $t(C_1)$ and $t(C_2)$ in separate parts of the bipartition, then for the remaining connected components C of $G(M) - S$ add $t(C)$ to either part. This defines a bipartition where the shared character states (known as the *splitting vector* [13]) are exactly the vertices of S. This implies that $|\Delta^L_{G(M)}| = O(m)$.

Let g be a proper cluster with splitting vector x, and S_x be the vertices of $G(M)$ corresponding to the character states in x. Define the equivalence relation g/x by the transitive closure of the relation tRt' if and only if there is a character χ^i where $\chi^i(t) = \chi^i(t')$ and $(\chi^i, \chi^i(t))$ is not a shared character state in x; calculating g/x takes $O(nm)$ time [12]. Given an equivalence class $[t]$ of g/x, the vertices $\{v \notin x \mid \text{ some } t' \in [t] \text{ is witnessed by } v\}$ are a connected component of $G(M) - S_x$, and every connected component can be described in this way. For a connected component C of $G(M) - S_x$, the size of its neighborhood can be calculated using the $t(C)$ rows of M (i.e. for $t \in t(C)$, count the character states of $[t]$ also in x, being careful not to overcount). $S_x \in \Delta^L_{G(M)}$ if and only if there are distinct equivalence classes $[t]$ and $[t']$ that share all character states in x. For each equivalence class, we examine each taxon once, so this requires a single pass through every row of M and can be done in $O(nm)$ time per proper cluster, so step 1 takes $O(nm^2)$ time.

For step 2, we check that all monochromatic pairs are separated in order to answer the decision question before construction proceeds. When S is witnessed by a proper cluster g with splitting vector x, we calculate the $S-$partition (this partition is exactly g/x). The $S-$partition will be useful in calculating the crossing relations, so the $S-$partition is stored when S is generated. There are $O(m)$ legal minimal separators and computing g/x takes $O(nm)$ time, so this calculation takes $O(nm^2)$ total time. Now, for each character there are 3 possible monochromatic pairs, and a monochromatic pair χ^i_j, χ^i_k are separated by S iff both character states are not in S and there is a pair of taxa s and t where $s_i = \chi^i_j$ and $t_i = \chi^i_k$ that lie in different pieces of the $S-$partition. This takes constant time to check per character and there are $O(m)$ characters, so it takes $O(m^2)$ time to determine if all monochromatic pairs are separated.

Before analyzing step 3, we first state two structural lemmas on minimal separators; the second follows from a lemma in [2].

Lemma 5. *[17] Let S and T be non-parallel minimal separators. Then for each full component C of $G - T$, S has a vertex in C.*

Lemma 6 (Lemma 3.10, [2]). *Let S and T be two minimal separators of a graph G. Then S and T are parallel if and only if there exists a full component C_S of $G - S$ and a connected component C_T of $G - T$ such that $C_S \subseteq C_T$.*

Because of the slight change from Lemma 3.10 in [2] and for completeness, we give a proof of Lemma 6.

Proof. Suppose S and T are parallel. Since S is a minimal separator, there are at least two full components in $G - S$ and since T is parallel to S, there is a full component C_1 of $G - S$ that does not intersect T. C_1 is connected in $G - T$, so there is a connected component C of $G - T$ containing C_1.

Now, suppose there are C_S and C_T satisfying the conditions of the lemma. Then $S \subseteq N(C_S) \subseteq N(C_T) \subseteq C_T \cup T$, implying S and T are parallel. □

Theorem 11. *There is an $O(nm^2)$ algorithm to calculate the crossing relations of $\Delta^L_{G(M)}$.*

Proof. Let $S, T \in \Delta^L_{G(M)}$. We begin by showing that S and T are parallel if and only if there is a full component C of $G(M) - S$ and connected component C' of $G(M) - T$ such that $t(C) \subseteq t(C')$.

Suppose S and T are parallel. From Lemma 6, there are connected components C of $G(M) - S$ and C' of $G(M) - T$ such that $C \subseteq C'$ and consequently $t(C) \subseteq t(C')$. Conversely, suppose that S and T are not parallel. Let C be a full component of $G(M) - S$ and C_T be a full component of $G(M) - T$. By Lemma 5, there is a vertex $v \in C \cap T$, and because C_T is full, there is a $u \in C_T \cap N(v)$. The taxa form an edge clique cover for $G(M)$, so there is a taxon t having both character states corresponding to u and v. Note $v \in C$ so $t \in t(C)$ and $u \in C_T$ so $t \in t(C_T)$. T has at least two full components, so there is another full component $C'_T \neq C_T$ of $G(M) - T$ such that $t(C) \cap t(C'_T) \neq \emptyset$. Thus $t(C) \not\subseteq t(C')$ for any connected component C' of $G(M) - T$.

It suffices to check for each full component C of $G(M) - S$ and connected component C' of $G(M) - T$ if $t(C) \subseteq t(C')$. There are $O(m^2)$ pairs of legal minimal separators, and this check takes $O(n)$ time ($O(nm^2)$ time overall) when the S-partition has been calculated. □

Recall that there are $O(m)$ minimal separators. Thus step 4 takes $O(m^2)$ time. For a graph G and a set of vertices X, let $G[X]$ denote the subgraph of G induced by vertices in X.

Lemma 7. *Let H be a proper triangulation of $G(M)$. Then H has at most $n-1$ minimal separators.*

Proof. Suppose H is a proper triangulation of $G(M)$. We proceed by induction, with the trivial base case $n = 1$ ($G(M)$ is a clique, and itself is the only proper triangulation). Let S be a minimal separator of H that does not contain any other minimal separator. Suppose the connected components of $G(M) - S$ are $C_1, C_2, \ldots, C_k$. Then the minimal separators of H are partitioned by S along with the minimal separators of $H[C_i \cup S]$ for $i = 1, 2, \ldots, k$ (see [11]). The subsets of taxa $t(C_i)$ for $i = 1, 2, \ldots, k$ partition the taxa defined by M. Let M_i be the submatrix of M induced by $t(C_i)$. Then $H[C_i \cup S]$ is a proper triangulation for the partition intersection graph $G(M_i)$. By induction, $H[C_i \cup S]$ has at most $|t(C_i)| - 1$ minimal separators, so H has at most $1 + \sum_{i=1}^{k}(|t(C_i)| - 1) \leq n - 1$ minimal separators. □

Each legal minimal separator has fewer than m vertices [10]. This fact along with Lemma 7 gives the following result.

Theorem 12. *Suppose that Q is a maximal pairwise parallel set of legal minimal separators of $G(M)$, and G_Q is a proper triangulation of $G(M)$. Then G_Q has $O(n)$ minimal separators, $|V(G_Q)| = O(m)$, and $|E(G_Q)| = O(m^2)$. The set $E(G_Q) - E(G(M))$ can be calculated in $O(nm^2)$ time by saturating each minimal separator in Q.*

Theorem 12 shows that step 5 takes $O(nm^2)$ time, and that $O(|V(G_Q)| + |E(G_Q)|) = O(m^2)$ so using MCS in step 6 takes $O(m^2)$ time. Thus our minimal separator algorithm for constructing perfect phylogenies for $r = 3$ is competitive with the algorithm of Dress and Steel [7], giving our main result.

Theorem 13. *The algorithm Proper Triangulation for 3-State Characters is* $O(nm^2)$.

4 Conclusions and Further Work

We have demonstrated how to use the minimal separator approach introduced in [10] to construct a perfect phylogeny for 3-state data in $O(nm^2)$ time. Attempting to pursue a time bound faster than $O(nm^2)$ raises a number of questions and directions for future work. Naively, the number of edges of the partition intersection graph and the number of F-edges are $O(m^2)$. Is this bound tight if M is a set of 3-state taxa with a perfect phylogeny? If so, any approach that utilizes MCS as a clique tree construction method will not improve our time bound. This would imply that explicit analysis of the edges of the partition intersection graph (or a proper triangulation of the partition intersection graph) is inadequate to study a speedup, and all computations would be done using M (instead of $G(M)$) aided with theoretical results about $G(M)$. Constructing tree representations in order to minimally triangulate a graph without explicitly computing the fill edges was studied in [2] in order to achieve a faster time bound, and it would be interesting to see if these ideas can be extended to find a faster construction algorithm for 3-state perfect phylogeny.

Acknowledgments. This research was partially supported by NSF grants CCF-0515378, IIS-0803564, and CCF-1017580.

References

1. Agarwala, R., Fernandez-Baca, D.: A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. SIAM Journal on Computing 23, 1216–1224 (1994)
2. Berry, A., Bordat, J.-P., Heggernes, P., Simonet, G., Villanger, Y.: A wide-range algorithm for minimal triangulation from an arbitrary ordering. Journal of Algorithms, 33–66 (2006)
3. Blair, J.R.S., Peyton, B.W.: An introduction to chordal graphs and clique trees. IMA Volumes in Mathematics and its Applications 56, 1–27 (1993)
4. Bodlaender, H., Fellows, M., Warnow, T.: Two strikes against perfect phylogeny. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 273–283. Springer, Heidelberg (1992)
5. Buneman, P.: A characterization of rigid circuit graphs. Discrete Math. 9, 205–212 (1974)
6. Dirac, G.A.: Abh. Math. Sem. Univ. Hamburg (1961)
7. Dress, A., Steel, M.: Convex tree realizations of partitions. Applied Math. Letters 5, 3–6 (1993)
8. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Annals of Discrete Mathematics, vol. 57. North-Holland Publishing Co., Amsterdam (2004)

9. Gusfield, D.: Efficient algorithms for inferring evolutionary trees. Networks 21, 19–28 (1991)
10. Gusfield, D.: The multi-state perfect phylogeny problem with missing and removable data. J. Computational Biology (preliminary version in Research in Computational Molecular Biology (RECOMB 2009)), 383–399 (2010)
11. Ho, C.W., Lee, R.C.T.: Efficient parallel algorithms for finding maximal cliques, clique trees, and minimum coloring on chordal graphs. Inf. Process. Lett. 28, 301–309 (1988)
12. Kannan, S., Warnow, T.: Inferring evolutionary history from DNA sequences. SIAM J. on Computing 23, 713–737 (1994)
13. Kannan, S., Warnow, T.: A fast algorithm for the computation and enumeration of perfect phylogenies. SIAM Journal on Computing 26, 1749–1763 (1995)
14. Kloks, T., Kratsch, D., Spiurad, J.: Treewidth and pathwidth of cocomparability graphs of bounded dimension. Technical Report, Eindhoven University (1993)
15. Lam, F., Gusfield, D., Sridhar, S.: Generalizing the four gamete condition and splits equivalence theorem: Perfect phylogeny on three state characters. To appear in SIAM J. Discrete Mathematics (2011)
16. Parra, A., Scheffler, P.: How to use the minimal separators of a graph for its chordal triangulation. In: Fülöp, Z. (ed.) ICALP 1995. LNCS, vol. 944, pp. 123–134. Springer, Heidelberg (1995)
17. Parra, A., Scheffler, P.: Characterizations and algorithmic applications of chordal graph embeddings. Discrete Applied Mathematics, 171–188 (1997)
18. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, Oxford (2003)
19. Steel, M.A.: The complexity of reconstructing trees from qualitative characters and subtrees. Journal of Classification 9, 91–116 (1992)
20. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM J. Comput. 13, 566–579 (1984)

On a Conjecture about Compatibility of Multi-states Characters

Michel Habib and Thu-Hien To

Université Paris Diderot - Paris 7, LIAFA, Case 701 75205 Paris Cedex 13, France
{habib,toth}@liafa.jussieu.fr

Abstract. Perfect phylogeny consisting of determining the compatibility of a set of characters is known to be NP-complete [4,28]. We propose in this article a conjecture on the necessary and sufficient conditions of compatibility: Given a set $\mathcal{C}$ of r-states full characters, there exists a function $f(r)$ such that $\mathcal{C}$ is compatible iff every set of $f(r)$ characters of $\mathcal{C}$ is compatible. According to [7,9,8,25,11,23], $f(2) = 2$, $f(3) = 3$ and $f(r) \geq r$. [23] conjectured that $f(r) = r$ for any $r \geq 2$. In this paper, we present an example showing that $f(4) \geq 5$. Therefore it could be the case that for $r \geq 4$ characters the problem behavior drastically changes. In a second part, we propose a closure operation for chordal sandwich graphs. The later problem is a common approach of perfect phylogeny.

Keywords: perfect phylogeny, characters compatibility, chordal sandwich graph, vertex-coloured graph, triangulation.

1 Introduction

Given an input biological data of a currently-living species set, phylogenetics aims to reconstruct evolutionary history of their ancestors. The evolutionary model of perfect phylogeny is phylogenetic tree, and the data are characters of species. Characters can be morphological, biochemical, physiological, behavioural, embryological, or genetic. Each character has several states. Here are some examples. The character *have wings* has two states: with wings and without wings. The character *number of legs* has many states: one leg, two legs, four legs, ... These are morphological characters. For an example of genetic characters, given a set of DNA sequences having a same length, if we consider each position on the sequences to be a character, then each character has 4 states corresponding to 4 bases of DNA as *A, T, C, G*.

Let $\mathcal{L}$ be a species set, and let c be a character on $\mathcal{L}$. Then, c can be represented by a partition of a non-empty subset $\mathcal{L}'$ of $\mathcal{L}$ such that each part consists of all species having the same state of c. So a set of characters is a set of partitions. A character is said to be trivial if the partition has at most one part having more than 1 element. Otherwise, it is non-trivial. If $\mathcal{L}' = \mathcal{L}$, then c is a full character, otherwise it is a partial character. If c has at most r parts, then c is an r-states character. A binary character is a 2-states full character.

A phylogenetic tree on a species set $\mathcal{L}$ is a tree in which each leaf is labelled distinctly by a species of $\mathcal{L}$.

Definition 1. *[6] Let c be an r-states character and let T be a phylogenetic tree on $\mathcal{L}$. For $i = 0, \ldots, r-1$, denote by $T_i(c)$ the minimal subtree of T on the leaf set consisting of the species having the state i of c. So, c is said to be* ***convex*** *on T iff the subtrees $T_i(c)$ are pairwise vertex-disjoint.*

A set of characters is **compatible** iff there exists a phylogenetic tree on which every character is convex. The character compatibility problem is also known as the **perfect phylogeny** problem.

Example 1. Let $\mathcal{L} = \{a, b, c, d, e, f, g\}$ and $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ be a collection of characters on $\mathcal{L}$ such that: $C_1 = ab|cdefj|ghi$, $C_2 = def|abcghij$, $C_3 = gh|defi$, $C_4 = abcd|ghi$. So, $\mathcal{C}$ is compatible because there is a phylogenetic tree T in Figure 1 on which every character is convex. For example, C_1 is convex on this tree because the subtrees of T on $\{a, b\}$, $\{c, d, e, f, j\}$ and $\{g, h, i\}$ are pairwise vertex-disjoint. Similarly, C_2, C_3, and C_4 are also convex of this tree.

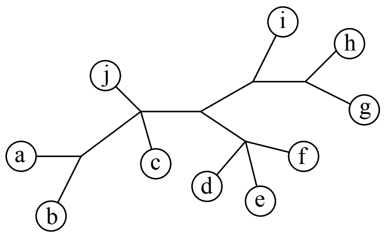


Fig. 1. A phylogenetic tree on which all characters of $\mathcal{C}$ are convex

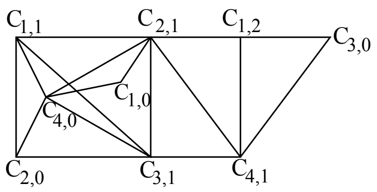


Fig. 2. The partition intersection graph of $\mathcal{C}$

Some previous works

Existence of perfect phylogeny: Given a set of characters on $\mathcal{L}$, is there any phylogenetic tree on which all characters are convex? It is easy to determine whether a collection of binary characters is compatible. There are polynomial algorithms for checking compatibility of 3-states full characters [6] and 4-states full characters [21]. In general, there are polynomial algorithms in the number of characters and species, but exponential in the number of states [24,1,22,2]. However the problem is NP-complete even for 2-states characters [4,28]. There exists effective, practical approaches for 2-states characters [17], and a new approach for this problem is proposed in [13].

Quartet problem: A minimal non-trivial character is a 2-states character such that each state contains exactly two species. Such character is called a *quartet*. As stated in the previous paragraph, the problem of compatibility of a set of quartets is NP-complete [4,28]. However, there are some particular cases that the problem is polynomial [3], see [26] for details.

Define a tree by characters: a set of characters defines a tree iff there is not any other tree on which these characters are convex. [27] showed that a set of 3 characters are not sufficient to define a tree but a set of 5 characters are. Later, [20] showed that for any tree, there exist at most 4 characters which define this tree. Hence, 4 is the optimal value. For the problem of whether a set of characters defines a tree, this is recently proved to be NP-hard [18].

Maximum parsimony: When there is no perfect phylogeny that can be inferred from data, it is desirable to find a model that minimize the number of reverse and convergent transitions. That is the problem of maximum parsimony.

Perfect phylogeny with recombination: When the characters set are not tree-representable, it is also interesting to construct a model that can represent their evolution. The model used here is recombination networks. Introduced by [19], intensive work have been done since then on this problem, including [29,16,15,12,14].

In this article, we are interested in the necessary and sufficient conditions of compatibility of a set of r-states full characters. We propose the following conjecture.

Conjecture: For any set $\mathcal{C}$ of r-states full characters, there exists a function $f(r)$, which does not depend on n, such that $\mathcal{C}$ is compatible iff every set of $f(r)$ characters of $\mathcal{C}$ is compatible.

This conjecture is based on the following previous results. According to [9,25,11,23], $f(2) = 2$ and $f(3) = 3$. There are polynomial algorithms in the number of characters and species, but exponential in the number of states [24,1,22,2].

2 Preliminaries

A very popular approach for studying perfect phylogeny is to consider chordal completion of vertex-coloured graphs, or equivalently chordal sandwich graph problems.

Definition 2. *[5] Let $\mathcal{L} = \{x_1, \ldots, x_m\}$ be a species set and let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a set of characters on $\mathcal{L}$. Each C_i is a partition of a subset of $\mathcal{L}$. The* ***partition intersection graph*** *$G = (V, E)$ of $\mathcal{C}$ is constructed as follows:*

- *Each character of $\mathcal{C}$ is associated with a different colour.*
- *Each vertex of V corresponds to a state of a character of $\mathcal{C}$. This vertex is then coloured by the colour of the character.*
- *There is an edge between 2 vertices if the 2 corresponding states of the 2 characters have at least a common species.*

In our figures, instead of colouring the vertices, we include the name of the characters in the labels of the vertices. For example, let consider the character set in Example 1. The partition intersection graph of $\mathcal{C}$ is in Figure 2. Each vertex $C_{i,j}$ represents the state j of character i.

A graph G is *chordal* if every cycle of length ≥ 4 contains at least a chord. A *chordal completion* of G is a chordal graph $G' = (V, E')$ such that $E \subseteq E'$.

This completion is minimal iff when we remove any edge in $E' \setminus E$, the resulting graph is not chordal. Given a vertex-coloured graph G, a **proper chordal completion** of G is a chordal graph $G' = (V, E')$ such that $E \subseteq E'$ and E' does not contain any edge connecting two vertices of the same colour.

Theorem 1. *[5,25,28] A set of characters $\mathcal{C}$ is compatible iff its partition intersection graph has a proper chordal completion.*

The set of characters in Example 1 is compatible, its partition intersection graph (Figure 2) has indeed a proper chordal completion which is itself.

Proper chordal completion of vertex-coloured graph can be stated equivalently under the form of sandwich problems, which were introduced by Golumbic, Kaplan and Shamir in [10] for DNA physical mapping problems.

Definition 3. *Let us denote by $G = [V, E, F]$ the sandwich problem on a graph $G = (V, E)$, where F is a set of impossible edges such that $E \cap F = \emptyset$.*

*If there is a graph $G_S = (V, E_S)$ such that $E \subseteq E_S \subseteq V \times V \setminus F = \overline{F}$ and G_S satisfies property Π, then G_S is called a **Π-sandwich graph** of G.*

It is easy to see that a chordal completion of a vertex-coloured graph $G = (V, E)$ is proper iff it is a chordal-sandwich graph of $G = [V, E, F]$ where F is the set of pairs of vertices having a same colour. So, by considering this set F, we can ignore the colours of the initial graph. We also call a *chordal-sandwich graph* of G a *proper chordal completion* of G, i.e. a chordal completion of G without using any edge belonging to F.

3 Our Contributions

Fitch-Meacham examples first introduced in [7,8], then later generalized in [25] and formally proved in [23], showed that $f(r) \geq r$ for any $r \geq 2$. [23] conjectured that for any r, there is a perfect phylogeny on r-state characters if and only if there is one for every subset of r characters, i.e. $f(r) = r$ for any $r \geq 2$. However, we propose an example in the next Section which shows that $f(4) \geq 5$. It improves Fitch-Meacham's lower bound and disproves the conjecture in [23]. **Therefore it could be the case that for $r \geq 4$ characters the problem behavior drastically changes**. After that, in Section 5, we propose a closure chordal sandwich graph operation such that the obtained graph has a stronger structure.

4 The 4-States Characters Case

We present here an example of a set of 4-states characters which is not compatible, but every 4 characters subsets are compatible.

Let $\mathcal{C}$ be the following set of characters:
$A = \{x, u\}|\{z, t\}|\{y\}|\{v\} = A_0|A_1|A_2|A_3$
$B = \{x, y\}|\{t, v\}|\{z\}|\{u\} = B_0|B_1|B_2|B_3$

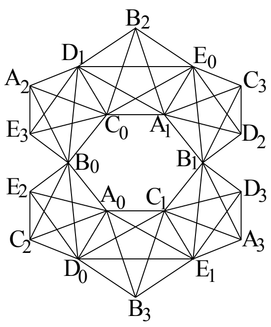


Fig. 3. Graph G

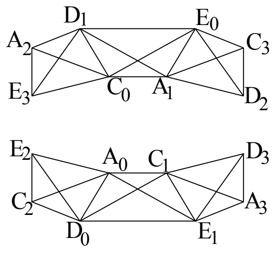


Fig. 4. The induced subgraph of G on 4 colours A, C, D, E

$C = \{y, z\}|\{u, v\}|\{x\}|\{t\} = C_0|C_1|C_2|C_3$
$D = \{x, u\}|\{y, z\}|\{t\}|\{v\} = D_0|D_1|D_2|D_3$
$E = \{z, t\}|\{u, v\}|\{x\}|\{y\} = E_0|E_1|E_2|E_3$
Each character has 4 states that we denote by A_0, A_1, A_2, A_3 for the character A. The partition intersection graph G associated to $\mathcal{C}$ is in Figure 3.

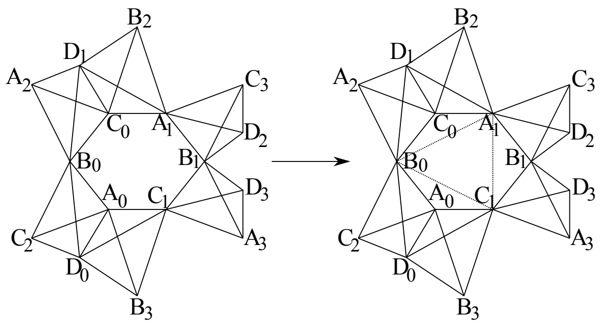


Fig. 5. The induced subgraph of G on 4 colours A, B, C, D and its chordal completion

G does not accept any proper chordal completion. Indeed, if we consider only the induced subgraph of G on 4 colours A, B, C, D and triangulate it, then there is a unique way to do that by connecting $(A_1, B_0), (B_0, C_1)$ and (C_1, A_1) (Figure 5). Similarly, if we consider the induced subgraph of G on 4 colours A, B, C, E, there is also a unique way to triangulate it by connecting $(A_0, B_1), (B_1, C_0)$ and (C_0, A_0). So, to triangulate G, the cycle $[A_0, B_1, A_1, B_0, A_0]$ is forced to be created. However, this cycle has no proper chordal completion, so G has no proper chordal completion, i.e. $\mathcal{C}$ is not compatible.

However, as we see in Figure 4, the induced subgraph of G on 4 colours A, C, D, E is chordal. The induced subgraph of G on 4 colours A, B, C, D has a proper chordal completion (Figure 5) and similarly for the induced subgraph of G on 4 colours A, B, C, E due to the symmetry. The induced subgraph of G on 4

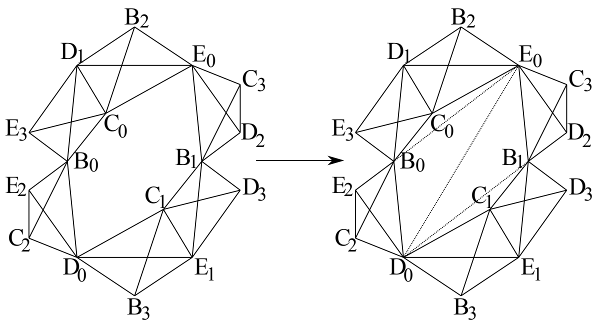


Fig. 6. The induced subgraph of G on 4 colours B, C, D, E and its chordal completion

colours B, C, D, E also has a proper chordal completion (Figure 6) and similarly for the induced subgraph of G on 4 colours A, B, D, E.

It means that every 4 characters of $\mathcal{C}$ are compatible but the whole set $\mathcal{C}$ is not compatible.

5 A Closure Operation for Chordal Sandwich Problems

Given a chordal sandwich problem $G = [V, E, F]$ where $E \cap F = \emptyset$, let u, v be two vertices of G such that $(u, v) \notin E$.

(u, v) is an *impossible edge* if it is not included in any minimal proper chordal completion of G. So (u, v) is impossible if either $(u, v) \in F$ or if by connecting them, the resulting graph does not have any proper chordal completion. The impossible edges are represented by dashed lines in our figures.

(u, v) is a *forced edge* if it is contained in every proper chordal completion of G. So if there is a cycle in G which has a unique proper chordal completion, then the edges used to complete this cycle are forced.

A cycle C of G is *impossible* if every chordal completion of C contains at least an edge in F. So if G has a proper chordal completion then it has no impossible cycle. The converse is not always true. For example, see the graph $G = [V, E, F]$ in Figure 7(b) where F consists of the pairs of vertices having a same colour. This graph has 3 chordless cycles and each one can be chordally completed without using any edge in F. However, G does not admit any proper chordal completion.

Example 2. In Figure 7(a) we have a cycle of size 5 on 3 colours a, b, c. The set F consists of (a_0, a_1) and (b_0, b_1). One can deduce that (a_1, b_1) is impossible since by connecting them we have the impossible cycle $[a_0, b_0, a_1, b_1, a_0]$. We deduce furthermore that (c_0, a_0) and (c_0, b_0) are forced because the unique way to properly chordally complete this cycle is by connecting them.

Consider the graph in Figure 7(b) where F is the set of pairs of vertices having a same colour. Similarly to the previous example on the cycle $[a_0, b_1, c_0, a_1, b_0, a_0]$, we deduce that c_0a_0 and c_0b_0 are forced. So, the cycle $[b_0, c_0, b_2, c_1, b_0]$ is forced to be present in any proper chordal completion of this graph. However, this cycle is impossible. Therefore, this graph does not have any proper chordal completion.

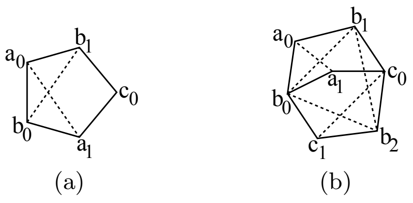


Fig. 7. Example 2

Note that by adding any forced edge into E or any impossible edge into F, we do not lose any proper chordal completion. Therefore we can introduce a 1-closure operation on sandwich problems by considering the effect on an edge in the problem, more formally:

Definition 4. *A chordal sandwich problem $G = [V, E, F]$ where $E \cap F = \emptyset$, is **1-closed** if:*

a) by connecting any pair of vertices not in $E \cup F$, there is no impossible cycle created.

b) any cycle of $G = (V, E)$ has at least two minimal proper chordal completions.

Observation 1. *Given a cycle $C = [u_1, \ldots, u_k, u_1]$, then:*

(i) for any $i \in \{1, \ldots, k\}$, every chordal completion of C must contain either (u_{i-1}, u_{i+1}) or (u_i, u_j) for a certain j different from $i, i-1, i+1$.

(ii) every chordal completion of C must contain a chord (u_{i-1}, u_{i+1}) for a certain i.

Lemma 1 (Detecting impossible edges and forced edges)

Let $G = [V, E, F]$ be a chordal sandwich problem where $E \cap F = \emptyset$ and (u, v) be two vertices of G:

1) If there is a chordless path $[u, t_1, \ldots, t_k, v]$ such that for every $i = 1, \ldots, k$, either (u, t_i) or (v, t_i) is impossible, then (u, v) is also impossible (Figure 8).

2) Suppose that $(u, v) \in E$. If there is a chordless cycle $C = [u, w, t_1, \ldots, t_k, v, u]$ such that for every $i = 1, \ldots, k$, either (u, t_i) or (v, t_i) is impossible, then (v, w) is a forced edge (Figure 9).

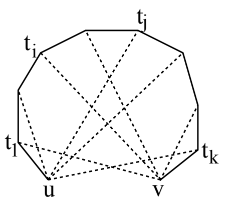


Fig. 8. (u, v) is impossible

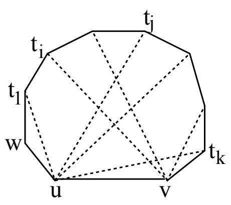


Fig. 9. (v, w) is a forced edge

Proof: 1) By connecting (u, v), we obtain the chordless cycle $C = [u, t_1, \ldots, t_k, v, u]$. We will prove that C is impossible. By Observation 1-(i), to chordally complete C, we must connect either (t_1, v) or (u, t_i) for a certain $i = 2, \ldots, k$. However, (t_1, v) is impossible because $(u, t_1) \in E$ and by the assumption, either (u, t_1) or (v, t_1) must be impossible. So, we must connect an edge (u, t_i), t_i not t_1, which must not be an impossible edge. We deduce that (v, t_i) is impossible. The created chordless subcycle $[u, u_i, u_{i+1}, \ldots, u_k, v, u]$ has the same property as C. So, by using the same argument, to chordally complete this cycle, we must connect an edge (u, u_j) where $i < j \leq k$. The size of the considering cycle strictly decreases each time we apply this argument. Continuing with this reasoning, eventually u is forced to connect to t_k, but (u, t_k) is impossible because $(v, t_k) \in E$, a contradiction.

2) We will prove that, every proper chordal completion of C must contain the chord (v, w). Suppose that there is a proper chordal completion of this cycle which does not contain (v, w). By Observation 1-(i), this completion must contain (u, t_i) for a certain $i = 1, \ldots, k$. We obtain then the chordless subcycle $[u, t_i, t_{i+1}, \ldots, t_k, v, u]$. The path $[u, t_i, t_{i+1}, \ldots, t_k, v]$ satisfies the condition of 1), so (u, v) is an impossible edge. However, $(u, v) \in E$, so this subcycle is impossible. In other words, C does not have any proper chordal completion which does not contain (v, w). Hence, (v, w) must belong to every chordal completion of G, i.e. it is a forced edge. □

Corollary 1. *Let $G = [V, E, F]$ be a chordal sandwich problem, and denote by $F(u)$ the set of vertices u' such that $(u, u') \in F$. Then, for every pair of vertices (u, v):*

1) If (u, v) is not an edge of G and there is a chordless path $[u, t_1, \ldots, t_k, v]$ such that for every $i = 1, \ldots, k$, $t_i \in F(u) \cup F(v)$ then (u, v) is impossible. We call such a path a $f(u, v)$ path.

2) If (u, v) is an edge of G and there is a chordless cycle $[u, w, t_1, \ldots, t_k, v, u]$ such that $w \notin F(v)$ and for every $i = 1, \ldots, k$, $t_i \in F_u \cup F_v$, then (v, w) is a forced edge. We call such a cycle a $g(u, v, w)$ cycle.

Theorem 2. *Algorithm 1 takes time $O(n^4(n + m))$ and either answers "Yes there is a solution or that there is no solution", or the 1-closure of G.*

Proof: **Correctness:** According to Corollary 1, the loop *for* recognizes all pairs of vertices (u, v) which satisfy the conditions in the corollary to detect impossible edges and forced edges. So the pairs of vertices added in F' at line 8 are the impossible edges, i.e. the edges which are not included in any proper chordal completion of G. Furthers, the pairs of vertices added in E' at line 13 are forced edges, i.e. the edges which are included in every proper chordal completion of G. Therefore, any proper chordal completion of G' is a proper chordal completion of G and vice versa.

Moreover, if (u, v) is an edge and there exists a $f(u, v)$ path then according to the proof of Lemma 1, the chordless cycle consisting of $f(u, v)$ and (u, v) is impossible. Hence, G does not have any proper chordal completion (line 11).

This process stops when there is no more impossible edges or forced edges detected. So, in G', for any pair of vertices (u, v), there is no $f(u, v)$ path; and if $(u, v) \in E'$, then there is no $g(u, v, w)$ cycle. We will prove that G' is the 1-closure of G, i.e. it satisfies the two properties in Definition 4.

a) The first property: we prove by induction on the size of cycles.

```
Data: A chordal sandwich problem $G = [V, E, F]$
Result: Answer Yes-No on the existence of a proper chordal completion,
        or the 1-closure of $G$
1  For any $u \in V$, calculate $N(u) = \{v |\ (u, v) \in E\}$ and
   $F(u) = \{v |\ (u, v) \in F\}$;
2  $E' = E$; $F' = F$; $flag = true$;
3  while ($flag$) do
4      $flag = false$;
5      for (any pair of vertices $(u, v)$) do
6          if (there is a $f(u, v)$ path) then
7              if ($u \notin N(v)$) then
8                  Add $u$ to $F(v)$, $v$ to $F(u)$, and $(u, v)$ to $F'$;
9                  $flag = true$;
10             else
11                 $G$ does not have any proper chordal completion; exit;
12         if ($u \in N(v)$) $\wedge$ (there is a $g(u, v, w)$ cycle) then
13             Add $v$ to $N(w)$, $w$ to $N(v)$, and $(u, v)$ to $E'$;
14             $flag = true$;
15 if ($G' = (V, E')$ is chordal) then
16     $G'$ is the unique chordal completion of $G$; return $G'$;
17 else if ($E' \cup F' = V \times V$) then
18     $G$ does not have any proper chordal completion; exist;
19 return $G' = [V, E', F']$
```

Algorithm 1. Computing the 1-closure of a chordal sandwich problem

Let (u, v) be a pair of vertices not in $E' \cup F'$. By connecting (u, v), let C be a created cycle which contains (u, v). We will prove that C admits at least a chordal completion without using any edge in F'.

For the case $|C| = 4$, let $C = [u, x, y, v, u]$. By the assumption, G' does not contain any $f(u, v)$ path, i.e. $[u, x, y, v]$ is not a $f(u, v)$ path. It means that either $(v, x) \notin F'$ or $(u, y) \notin F'$. So, we can complete C by connecting either (v, x) or (u, y).

Suppose that C has at least a chordal completion if $|C| \leq k$.

For the case $|C| = k+1$, let $C = [u, t_1, \ldots, t_{k-1}, v, u]$. Because $[u, t_1, \ldots, t_{k-1}, v]$ is not a $f(u, v)$ path in G', there exists at least an $i \in \{1, \ldots, k-1\}$ such that $(u, t_i), (v, t_i) \notin F'$. By the induction hypothesis, if we connect (u, t_i), (v, t_i), the two subcycles $[u, t_1, \ldots, t_i, u]$ and $[v, t_i, \ldots, t_k, v]$ have proper chordal completions without using any pair of vertices in F' because both these two cycles have size smaller than $k + 1$. Completing these two subcycles gives a proper chordal completion for C. So, C admits at least one chordal completion.

b) The second property: Let C be a chordless cycle of G'. So C has at least one proper chordal completion because otherwise Algorithm 1 returns No at line 11. By Observation 1-(ii), this chordal completion must contain at least a triangle (u, v, w) such that (u, v), (u, w) are edges of C. Let $C = [u, w, t_1, \ldots, t_k, v, u]$, so C is not a $g(u, v, w)$ cycle because otherwise (v, w) is a forced edge and it must have been connected by the algorithm, a contradiction with the fact that C is chordless. So, there is a t_i such that $(u, t_i), (v, t_i) \notin F'$. Using the first property, by connecting (u, t_i) and (v, t_i), we obtain two chordless subcycles which have proper chordal completions. That implies another chordal completion of C containing $(u, t_i), (v, t_i)$ and does not contain (v, w). So it is different with the initial one. In other words, C has at least 2 distinct minimal chordal completions without using any pair of vertices in F'.

Complexity

- Calculating $N(u)$ and $F(u)$ for any vertex u in line 1 is done in times $O(n^2)$.
- The loop *while*: For each iteration, there is at least a pair of vertices (u, v) modified, i.e (u, v) becomes either an impossible edge or a forced edge. This can only be done once. The loop stops when there is no more modification on any pair of vertices. So, the number of iterations of this loop is bounded by the number of pairs of vertices, i.e by $O(n^2)$.
- The loop *for*: there are $O(n^2)$ pairs of vertices (u, v). So there are $O(n^2)$ iterations. In each iteration:
  - Checking if there is a simple path $f(u, v)$ can be done in linear time: We proceed a Breadth First Search (BFS) starting at u such that the visited vertices are in $F(u) \cup F(v) \setminus N(u)$. If we meet a vertex in $N(v)$, then there is a $f(u, v)$ path. Otherwise, there is no such path.
  - Checking if there is a $g(u, v, w)$ cycle can also be done in linear time: We proceed a BFS from u such that the first visited vertices is not in $N(v) \cup F(v)$, and the remaining visited vertices are in $F(u) \cup F(v) \setminus N(u)$. If we meet a vertex in $N(v)$ then we have a $g(u, v, w)$ cycle. Otherwise, there is no such cycle.

So, the total complexity is $O(n^4(n + m))$ where n is the number of vertices of G' and m is the number of edges of the obtained graph. □

Although in most of our examples, computing the 1-closure is enough to decide character compatibility, this is not true in general. This 1-closure operation only reduces the size of the set $\overline{F} - E$ on which some other algorithm or method is required.

6 Conclusion

Our example in Section 4 showed that $f(4) \geq 5$. We suggest that $f(r) \geq r+1$ for any $r \geq 4$. So, a further work is to generalize this example. Another problem is to prove the existence of $f(r)$ by searching for an upper bound function $F(r)$ of $f(r)$, i.e. if every set $F(r)$ characters of $\mathcal{C}$ is compatible then $\mathcal{C}$ is compatible. A harder question is determining $f(r)$ for $r \geq 4$.

Acknowledgements. The authors wish to thank the referees for their insightful comments.

References

1. Agarwala, R., Fernández-Baca, D.: A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. SIAM J. Comput. 23(6) (1994)
2. Agarwala, R., Fernández-Baca, D.: Simple algorithms for perfect phylogeny and triangulating colored graphs. Int. J. Found. Comput. Sci. 7(1), 11–22 (1996)
3. Böcker, S., Dress, A.W.M., Steel, M.A.: Patching up x-trees. Annals of Combinatorics 3, 1–12 (1999)
4. Bodlaender, H.L., Fellows, M.R., Warnow, T.: Two strikes against perfect phylogeny. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 273–283. Springer, Heidelberg (1992)
5. Buneman, P.: A characterization of rigid circuit graphs. Discrete Mathematics 9, 205–212 (1974)
6. Dress, A., Steel, M.A.: Convex tree realizations of partitions. Applied Mathematics Letters 5(3), 3–6 (1992)
7. Fitch, W.M.: Toward finding the tree of maximum parsimony. In: Estabrook, G.F. (ed.) The Eighth International Conference on Numerical Taxonomy, pp. 189–220. W. H. Freeman and Company, San Francisco (1975)
8. Fitch, W.M.: On the problem of discovering the most parsimonious tree. American Naturalist 11, 223–257 (1977)
9. Johnson, C., Estabrook, G., McMorris, F.: A mathematical formulation for the analysis of cladistic character compatibility. Math Bioscience 29 (1976)
10. Golumbic, M.C., Kaplan, H., Shamir, R.: Graph sandwich problems. J. Algorithms 19(3), 449–473 (1995)
11. Gusfield, D.: Efficient algorithms for inferring evolutionary trees. Networks 21, 19–28 (1991)
12. Gusfield, D.: Optimal, efficient reconstruction of root-unknown phylogenetic networks with constrained and structured recombination. J. Comput. Syst. Sci. 70(3), 381–398 (2005)
13. Gusfield, D.: The multi-state perfect phylogeny problem with missing and removable data: Solutions via integer-programming and chordal graph theory. J. of Computational Biology 17(3), 383–399 (2010)
14. Gusfield, D., Bansal, V., Bafna, V., Song, Y.S.: A decomposition theory for phylogenetic networks and incompatible characters. Journal of Computational Biology 14(10), 1247–1272 (2007)
15. Gusfield, D., Eddhu, S., Langley, C.H.: The fine structure of galls in phylogenetic networks. INFORMS Journal on Computing 16(4), 459–469 (2004)

16. Gusfield, D., Eddhu, S., Langley, C.H.: Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. J. Bioinformatics and Computational Biology 2(1), 173–214 (2004)
17. Gusfield, D., Frid, Y., Brown, D.: Integer programming formulations and computations solving phylogenetic and population genetic problems with missing or genotypic data. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 51–64. Springer, Heidelberg (2007)
18. Habib, M., Stacho, J.: Unique perfect phylogeny is np-hard. In: Giancarlo, R., Manzini, G. (eds.) CPM 2011. LNCS, vol. 6661, pp. 132–146. Springer, Heidelberg (2011)
19. Hein, J.: Reconstructing evolution of sequences subject to recombination using parsimony. Mathematical Biosciences 98(2), 185–200 (1990)
20. Huber, K.T., Moulton, V., Steel, M.: Four characters suffice to convexly define a phylogenetic tree. SIAM Journal on Discrete Mathematics 18, 835–843 (2005)
21. Kannan, S., Warnow, T.: Inferring evolutionary history from dna sequences. SIAM J. Comput. 23(4) (1994)
22. Kannan, S., Warnow, T.: A fast algorithm for the computation and enumeration of perfect phylogenies when the number of character states is fixed. In: SODA, pp. 595–603 (1995)
23. Lam, F., Gusfield, D., Sridhar, S.: Generalizing the four gamete condition and splits equivalence theorem: Perfect phylogeny on three state characters. In: Salzberg, S.L., Warnow, T. (eds.) WABI 2009. LNCS, vol. 5724, pp. 206–219. Springer, Heidelberg (2009)
24. McMorris, F.R., Warnow, T., Wimer, T.: Triangulating vertex colored graphs. In: SODA, pp. 120–127 (1993)
25. Meacham, C.A.: Theoretical and computational considerations of the compatibility of qualitative taxonomic characters. In: Felsenstein, J. (ed.) Numerical Taxonomy. NATO ASI, vol. G1, pp. 304–314. Springer, Heidelberg (1983)
26. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, Oxford (2003)
27. Semple, C., Steel, M.: Tree reconstruction from multi-state characters. Advances in Applied Mathematics 28, 169–184 (2002)
28. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. Journal of Classification 9, 91–116 (1992)
29. Wang, L., Zhang, K., Zhang, L.: Perfect phylogenetic networks with recombination. Journal of Computational Biology 8(1), 69–78 (2001)

Learning Protein Functions from Bi-relational Graph of Proteins and Function Annotations

Jonathan Qiang Jiang[*]

Department of Computer Science, City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong
qiajiang@cityu.edu.hk

Abstract. We propose here a multi-label semi-supervised learning algorithm, PfunBG, to predict protein functions, employing a bi-relational graph (BG) of proteins and function annotations. Different from most, if not all, existing methods that only consider the partially labeled protein-protein interaction (PPI) network, the BG comprises three components, a PPI network, a function class graph induced from function annotations of such proteins, and a bipartite graph induced from function assignments. By referring to proteins and function classes equally as vertices, we exploit *network propagation* to measure how closely a specific function class is related to a protein of interest. The experiments on a yeast PPI network illustrate its effectiveness and efficiency.

Keywords: protein function prediction, multi-label learning, bi-relational graph, network propagation.

1 Introduction

Designing computational algorithms to predict protein function can be traced back to two decades ago. Traditional *in silico* protein function annotation mainly relied on collecting a set of features from each protein and applying machine learning algorithms, for example, the support vector machine (SVM), to get the classification [12]. With the advent of high-throughput experimental techniques (e.g., yeast two-hybrid and affinity purification with mass spectrometry), vast amounts of genome-scale protein interaction data have been produced and accumulated, which make it possible to discover gene/protein functions in the context of a network.

A protein-protein interaction (PPI) network is a weighted, undirected graph $G_P = (\mathcal{V}_P, \mathcal{E}_P, W_P)$, where $\mathcal{V}_P$ is the set of vertices (proteins), $\mathcal{E}_P$ is the set of edges (interactions), and $W_P \in \mathbb{R}^{n \times n}$ is the affinity matrix that indicates the reliability of protein interactions. For a protein function prediction problem, we have a set of proteins $\mathcal{P} = \{p_i\}_{i=1,\dots,n}$ and K functional categories $\mathcal{C} = \{c_i\}_{i=1,\dots,K}$. Suppose each of the first l proteins has a set of labels $\mathcal{Y}_i \subseteq \mathcal{F}$ represented by a binary vector $\mathbf{y}_i \in \{0,1\}^K$, such that $y_{ik} = 1$ if p_i belongs

[*] Corresponding author.

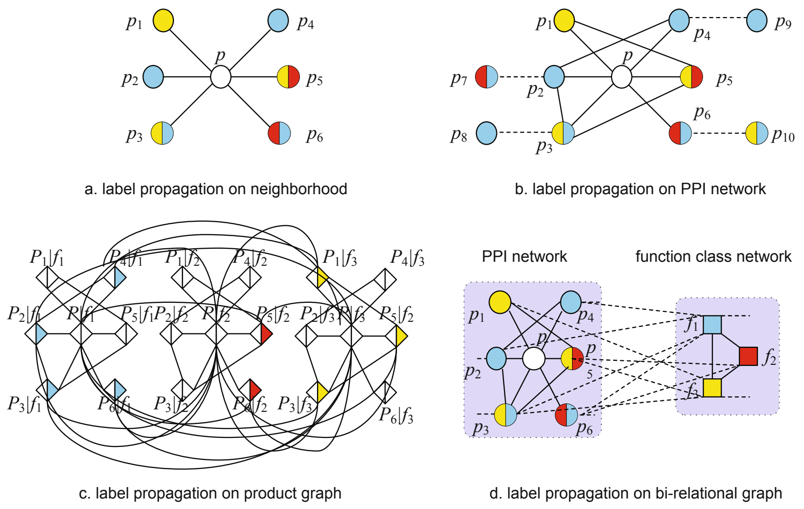

a. label propagation on neighborhood
b. label propagation on PPI network
c. label propagation on product graph
PPI network
function class network
d. label propagation on bi-relational graph

Fig. 1. Key principles in protein function annotation. Circle nodes, square nodes and diamond nodes represent proteins, function classes and node products, respectively. Annotated proteins are shown in different colors, the uncharacterized protein of interest is shown in white. We consider here assigning one function class (blue term) to the unannotated protein, which can be viewed as label propagation on different graphs. (a) local methods [7,13] only consider the functionality prevalent across the neighborhood of the investigated protein, i.e., a subgraph of the PPI network. (b) global approaches [14,9,10] take the full structure of network into account. (c) our previous algorithm, MCSL [8] can be understood as label propagation on a constrained product graph of PPI network and function class network. (d) The proposed algorithm in this paper studies label propagation on a bi-relational graph of proteins and function annotations.

to the k-th class, and 0 otherwise. Our goal is to predict the labels $\{\mathcal{Y}_i\}_{l+1}^{n}$ for the remaining unlabeled proteins. This problem, classifying nodes in a partially labeled graph, can be viewed as a graph-based supervised learning [19] or more intuitively, a label propagation process on such a graph [18]. Specifically, local methods (e.g., Majority[13] and χ^2-score [7]) transfer annotations among neighbor-nodes in the PPI network, assuming that nodes that are located close to each other tend to share the same function classes (Fig.1a). By contrast, global approaches take the full structure of the network into account (Fig.1b). Examples of such schemes include global assignment that minimizes the number of protein interactions between protein pairs that are annotated with different functions [14,9], and function assignment via propagating functions using links of the network [10].

However, most of the existing algorithms decompose the problem into K independent binary classification problems (one for each function class), and determine the labels for each protein by aggregating the classification results from all the classifiers [9,10]. Obviously, these algorithms ignore the inherent correlations among function classes, which could be an important hint for deciding the class memberships [3,17], and thus they suffer from the labeled data sparsity problem [8]. This phenomenon spurs the transition from multi-class learning to multi-label learning. The essential difference between these approaches is that classes in multi-class learning are assumed to be mutually exclusive while classes in multi-label learning are often correlated. To exploit the correlations among function classes, we recently built another graph, function class (FC) network $G_F = (\mathcal{V}_F, \mathcal{E}_F, W_F)$, where $\mathcal{V}_F$ is the set of vertices (function classes), $\mathcal{E}_F$ is the set of edges, and $W_F \in \mathbb{R}^{K \times K}$ is the affinity matrix that captures the functional similarity between each class, and proposed a new algorithm [8] that can effectively overcome the sparsity of label instance problem often suffer by previous approaches, and therefore significantly improve the prediction. It can be understood as a label propagation on a constrained product graph of PPI network and FC network (Fig.1c). Unfortunately, it may become time-consuming when large-scale interaction data are addressed.

Inspired by these works [8,16], we propose here a semi-supervised learning algorithm on a bi-relational graph that comprises three components, a PPI network, a function class network induced from the function annotations of such proteins, and a bipartite graph induced from the known function assignments. By referring to proteins and function classes equally as vertices, protein function prediction is reformulated as a problem of measuring how closely a specific function class is related to a given protein. Different from the work done by Wang et al. [16], we use *network propagation* to get this relevance. In addition, in the propagation process, all nodes of a cluster are not dealt with equally, as was done in [16]; instead, a particular node is weighted by the reliability of its interactions, compared with that of others nodes.

The rest of the paper is organized as follows. Section 2 presents the bi-relational graph model which our method builds on. We give the details of our algorithm in section 3 and experimentally evaluate the proposed approach in section 4. Finally, in section 5, we discuss and summarize our results.

2 Bi-relational Graph Model

Traditional graph-based protein function prediction methods only consider the PPI network G_P. In multi-label learning, the classes are interrelated to each other, and thus we built the FC network G_F that improves the predictive performance significantly [8]. In this paper, we plan to leverage both graphs.

The bi-relational graph of proteins and function annotations $G = (\mathcal{V}_p \cup \mathcal{V}_F, \mathcal{E}_P \cup \mathcal{E}_F \cup \mathcal{E}_R)$ contains three components, the PPI graph G_P, function class graph G_F, and a bipartite graph $G_R = (\mathcal{V}_P, \mathcal{V}_F, \mathcal{E}_R)$. G_R represents the known association between the proteins and function classes, and its adjacency matrix is $W_R \in \mathbb{R}^{n \times m}$. Semi-supervised learning on this graph can also be referred to as a label propagation process. For example, in Fig.1d, we aim to associate the uncharacterized protein p with a specific function class f_1 (blue term). We see clearly that this protein can receive not only the label information from four immediate neighbors, say, p_2, p_3, p_4 and p_6 which are annotated with function

f_1, but also the correlated label information from other partners which are assigned to function f_2 or f_3. This is the key idea underlying our previous work [8], in which we viewed the two types of vertices differently and thus constructed a constrained product graph. By contrast, in this paper, we treat protein vertices and function class vertices equally. In this way, assigning a specific function class to a given protein is equivalent to measuring the closeness between two vertices in the bi-relational graph.

Estimating the relative importance is a fundamental issue in web browsing, searching and navigation. One celebrated strategy is the personalized Pagerank [6], which mimics a surfer performing a random walk with restart (RWR) on the graph. The RWR model has been applied to different researches, ranging from multimedia cross-modal correlation [11], to protein complex prediction [2], to disease gene prioritization [4]. In particular, a recent computer vision research study further developed the RWR on bi-relational graphs for automatic image annotation and objective recognition [16]. Different from these previous works, we measure the closeness between vertices based on the simulation of network propagation. This idea is very similar to that of RWR, with one key difference. Namely, in network propagation, the flow of information is normalized not only by the total outgoing flow from each node, but also by the total incoming flow into each node. Therefore, our presented study can be viewed as a generalization of semi-supervised learning with local and global consistency [18] on the bi-relational graph.

3 Methods

Network propagation based models can be uniformly expressed in a regularized framework where the first term is a loss function to penalize the deviation from the given labels, and the second term is a regularizer to prefer label smoothness [18,3,17,8]. The key principle is the *consistent assumption* [18]: (1) nearby points are likely to share the same label; and (2) points on the same structure (typically referred to as a cluster or a manifold) are likely to have the same label. Algorithmically, let $\widetilde{Y} = [\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_n]^T$ where $\widetilde{y}_{ik}$ is the confidence score that the i-th data can be annotated with the k-th function. The model can be simulated iteratively as follows:

$$\widetilde{Y}(t+1) = (1-\alpha)P_{\mathrm{NP}}\widetilde{Y}(t) + \alpha Y \tag{1}$$

During each iteration, each vertex receives the information spreading from its neighbors (the first term), and also retains its initial information (second term). The parameter α specifies the relative amount of the information from its neighbors and its initial label information. Finally, the label of each unannotated protein is set to the function class of which it has received most information during the iteration process.

Propagation matrix. Following the work [18], the intra-subgraph propagation matrices P_P and P_F of G_P and G_F can be defined as

$$P_P = D_P^{-\frac{1}{2}} W_P D_P^{-\frac{1}{2}} \quad P_F = D_F^{-\frac{1}{2}} W_F D_F^{-\frac{1}{2}} \tag{2}$$

where D_P and D_F are the degree matrices of graph G_P and G_F. Similarly, the inter-subgraph propagation matrix P_{PF} between G_P and G_F are formulated as

$$P_{PF} = D_{RL}^{-\frac{1}{2}} W_R D_{RR}^{-\frac{1}{2}} \tag{3}$$

where

$$D_{RL} = diag(\sum_j W_R(1,j), \ldots, \sum_j W_R(n,j))^T \tag{4}$$

$$D_{RR} = diag(\sum_i W_R(i,1), \ldots, \sum_i W_R(i,K))^T \tag{5}$$

Inspired by [8,16], we design the propagation matrix P_{NP} on the bi-relational graph G as follows

$$P_{\text{NP}} = \begin{bmatrix} (1-\beta)P_P & \beta P_{PF} \\ \beta P_{PF}^T & (1-\beta)P_F \end{bmatrix} \tag{6}$$

where $0 \leq \beta \leq 1$ is a user-defined parameter that is used to adjust the relative importance of intra-subgraph and inter-subgraph information.

Label induced matrix. We can directly use the known association between proteins and function classes to feed the matrix Y in Eq.(1). But this strategy does not fully make use of the available information. Similar to [16,4], we think of both a function class vertex and its labeled training protein vertices as a function group

$$F_k = v_k^F \cup \{v_i^P | \mathbf{y}_{ik} = 1\}$$

In this way, instead of measuring vertex-to-vertex relevance between a function class vertex and an unannotated protein vertex, we measure the function group-to-vertex relevance between a function group and the protein. We build K label induced vectors, one for each function group $F_k (1 \leq k \leq K)$

$$\mathbf{Y}_k = \begin{bmatrix} \gamma Y_k^P \\ (1-\gamma) Y_k^F \end{bmatrix} \in \mathbb{R}_+^{n+K} \tag{7}$$

where $\mathbf{Y}_k^P(i) = 1/\sum_i y_{ik}$ if $y_{ik} = 1$ and $\mathbf{Y}_k^P(i) = 0$ otherwise; $\mathbf{Y}_k^F(i) = 1$ if $i = k$ and $\mathbf{y}_k^F(i) = 0$ otherwise. $\gamma \in [0,1]$ controls the relative importance of the protein vertices and function class vertex in the function group. One drawback of the above setting is that all protein vertices in a function group are treated equally. Actually, the importance of each protein vertex should not be the same. For example, if a particular vertex is known to be more reliable in the quality of its interactions than others, it should be weighted more than the others. Suppose that $w_v^{(k)}$ is the relative importance of node v in function group F_k. The weight of protein vertex v in function group is defined as follows $s_v^{(k)} = w_v^{(k)} / \sum_{u \in F_k} w_u^{(k)}$ if $v \in F_k$, and 0 otherwise. In this work, we use the function group degree of a protein vertex to represent its relative importance, i.e., $w_u^{(k)} = \sum_{v \in F_k} W_{uv}$.

Iterative algorithm. Suppose $\widetilde{Y}_P \in \mathbb{R}^{n \times K}$ denote the confidence score that the protein p_i can be annotated with the k-th function, and $\widetilde{Y}_F \in \mathbb{R}^{K \times K}$ is the vertex-to-vertex relevance in subgraph G_F, respectively. Let $\widetilde{Y} = [\widetilde{Y}_P; \widetilde{Y}_F] \in \mathbb{R}^{(n+K) \times K}$ and $Y = [\mathbf{Y}_1, \ldots, \mathbf{Y}_K]$, the equilibrium solution $\widetilde{Y}^*$ of Eq.(1) is determined by

$$\widetilde{Y}^* = [I - (1-\alpha)P_{\text{NP}}]^{-1} Y \tag{8}$$

To avoid computing the inverse matrix, similar to [8,18], we give the following iterative algorithm:

Algorithm 1. PfunBG:label propagation on a bi-relational graph

```
   Input: Affinity matrix of PPI network $W_P$, affinity matrix of FC network $W_F$,
          function annotation $\{\mathcal{Y}_i\}_1^l$, a pre-defined maximum iteration number
          max-iter, a pre-defined parameter $\alpha \in [0,1]$
   Output: function prediction $\{\mathcal{Y}\}_{l+1}^n$
1  Construct $P_{\text{NP}}$ by Eq.(2)–(6) and $Y$ by Eq.(7);
2  for $t = 1; t \leq \max - \text{iter}$ do
3      $\widetilde{Y}(t+1) = (1-\alpha)\widetilde{Y}(t) + \alpha Y$;
4      if Convergence then
5          Break;
6      end
7  end
8  Predict labels for $p_i$ using $\widetilde{Y}_P$ by adaptive decision boundary method [15];
```

4 Experiments and Results

Experiment setup. We construct the functional-linkage network using the protein interaction dataset compiled from BioGRID (release 3.1.73) [1]. In order to reduce the false positive rate, we used only those interactions that were confirmed by at least two publications. The largest connected component of such network consists of 3179 proteins with 12413 interactions. Nabieva et al. [10] show that different experimental sources of deriving PPI may have different reliability and the prediction results can be improved substantially when these differences in reliability are taken into consideration. Here, we follow the approach proposed by Nabieva et al. [10] and estimate the reliability of each experimental source by simply finding the fraction of interaction pairs from that source which shares at least one function. For each interaction between a pair of protein u and v, its reliability can be estimated as

$$r_{(u,v)} = 1 - \prod_{i \in E_{(u,v)}} (1 - r_i)^{n_{i,(u,v)}} \tag{9}$$

where r_i is the reliability of experimental source i, $E_{(u,v)}$ is the set of experimental sources in which interaction between protein u and v is observed, and $n_{i,(u,v)}$

is the number of times which interaction between u and v is observed from experimental source i. The function annotation scheme is taken from MIPS Funcat-2.1(data-20070316), which consists of 507 functional classes (FCs) arranged in hierarchical order. Note that a protein annotated with a FC is also annotated with all its super-classes. To avoid this bias, we consider the 68 second-level FCs. The key for building a function class network is deciding how to measure the similarity between different FCs. Intuitively, we use the data-driven manner via cosine similarity [8]

$$W_F(i,j) = \cos(\mathbf{z}_i, \mathbf{z}_j) = \frac{\langle \mathbf{z}_i, \mathbf{z}_j \rangle}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|} \tag{10}$$

where $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_K] = W_R$. The conventional classification performance metrics in statistical learning, *precision* and *F1 score*, are used to evaluate the proposed algorithms. For every class, the precision and F1 score are computed following the standard definition for a binary classification problem. To address the multilabel scenario, following [5], macro average and micro average of precision and F1 score are computed to assess the overall performance across multiple labels.

5-fold cross validation. We test the performance using 5-fold cross-validation, i.e, these yeast proteins are randomly divided into 5 groups, and each group, in turn, is separated from the original dataset and used for testing. In our implementation, we fixed the parameter $\alpha = 0.01$ as suggested by [18]. The trade-off parameters β and γ are selected through the validation process. We compare our algorithm to four state-of-the-art methods: (1) Majority approach (MA) [13], (2) GenMultiCut (GMC) [14,9], (3) FunctionFlow (Funflow) [10], and (4) multi-label correlated semi-supervised learning (MCSL) [8]. The results are summarized in Table 1. In Table 1, we see clearly that the algorithms, MCSL and PfunBG, which take the correlations among function classes, consistently, and sometimes significantly, outperforms the other three approaches that treat each class independently. Specifically, the metrics improve more than 30% on average. This indicates that the correlations are indeed a good hint for membership prediction in multi-label learning. Compared with our previous method MCSL, our new algorithm can achieve slightly better performance, and it is much more efficient due to reducing the order of propagation matrix from nK to $n + K$. We also weighted the protein vertex in a function group and applied it to the cross validation. As expected, this result, in further improvements of PfunBG-w. In addition, we implemented the iterative version of Pfun, similar to [16]. That is, during each iteration, we use the learned *Causal Relationships* to replace the affinity matrix of function class network. Unfortunately, from the results, PfunBG-iter in Table 1, we did not obtain the similar results as that shown in [16]. The macro average precision and the micro average precision is almost the same as that obtained by PfunBG-w. However, the macro average F1 score and the micro average F1 score are dramatically lower. This may be due to two reasons. One is that the PPI network has so much noise and false positives, and

Table 1. Classification performance comparison by 5-fold cross validations

Metrics	(%)	MA	GMC	FunFlow	MCSL	PfunBG	PfunBG-w	PfunBG-iter
Macro	Precision	38.87	42.77	40.09	62.91	64.12	**68.03**	68.01
	F1	32.33	30.43	31.32	71.07	77.54	**80.12**	33.12
Micro	Precision	45.48	54.66	51.88	65.18	73.82	**75.92**	74.47
	F1	46.77	47.18	46.87	76.25	79.24	**80.57**	20.27

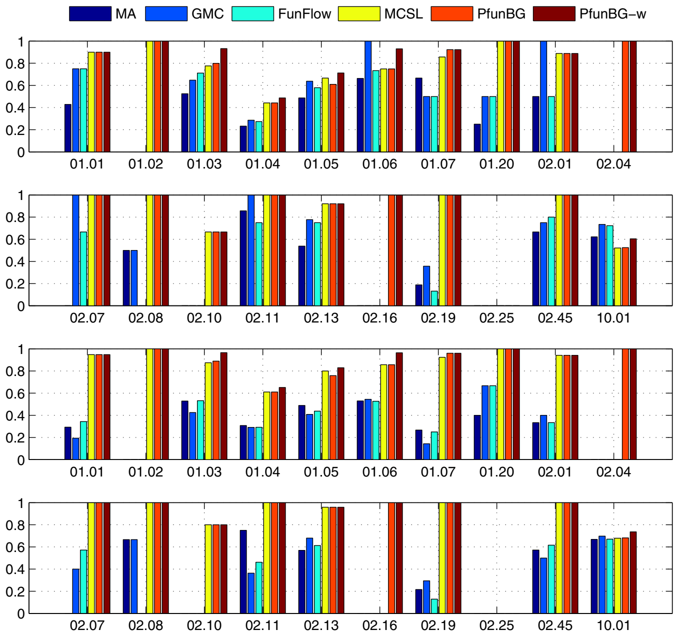


Fig. 2. The average precision and F1 score for the first 20 function classes

is not complete at the current time. The other is that the correlations among function classes are different from the correlation among semantic keywords. In particular, it is reported in [16] that the keywords can be coarsely split into two categories, objective class and background class. The relationship from an objective class to a background class is greater than that of the reverse. But in our case, it doesn't seem reasonable to refer to some function classes as objective classes and others as background classes.

Label propagation via label correlation. In order to explore why our new algorithm can improve performance significantly, we further check the average precision (AP) and F1 score for each function class. We observed that (1) when there are enough proteins annotated with a function class or when a function class is closely correlated with many other classes, all algorithms perform very well. Comparatively, the methods that take the correlations among function

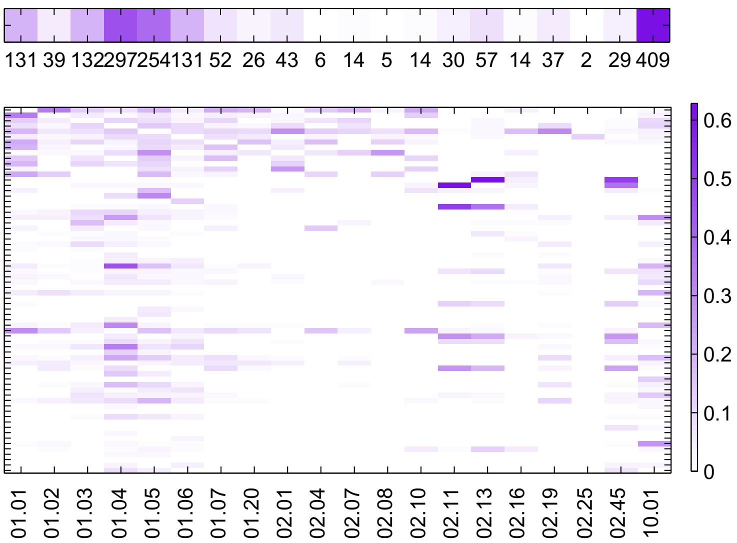


Fig. 3. The upper panel illustrates the number of proteins annotated with the first 20 function classes. The lower panel shows the correlations among the first 20 function classes.

classes into account can achieve better performance than the counterparts which treat each function class independently; (2) Only the methods that consider the correlations can successfully recover the annotations for a given function class when there is only a handful of proteins annotated with it, i.e., when labeled data sparsity occurs; (3) Only our new proposed algorithm can associate unlabeled protein with a specific function when there are few proteins annotated with it and when there is almost no correlation between this function class and the others. Specifically, a more careful examination is supplied in Fig. 2 where we give the AP and F1 score for the first 20 function classes. The number of proteins annotated with these function classes is illustrated as a bar in the upper panel of Fig.3. In addition, the data-driven correlations among the first 20 function classes and others are shown in the lower panel of Fig.3.

From these figures, we see clearly that function classes, $\{01.01, 01.03, 01.04, 10.01\}$ have many proteins labeled with them. Consequently, all algorithms perform very well even through our new algorithm always achieves superior performance. Although there are comparatively fewer proteins annotated with two functions $\{02.11, 02.13, 02.45\}$, they are closely correlated with each other (lower panel of Fig.3). Therefore, all the methods also can successfully recover the labels. However, the predictive performance changes significantly when the labeled data sparsity problem occurs. For the functions $\{01.02, 02.08, 02.19\}$, there are comparatively few proteins labeled with them, and they have loose correlations with other classes. Hence, all the approaches that treat each function class independently failed to recover the labels. As a benefit of taking the correlations among classes into account, MCSL and PfunBG still perform very well. As the case become much worse, say, for function classes $\{02.04, 02.16\}$, there are fewer proteins annotated with them, and they have much looser correlations with other function classes. None the approaches, except for our new algorithm, can assign these functions to a protein. All the observations indicate that the reason for the superior performance of our new algorithm is the additional label propagation via label correlation. This key idea can be effectively and efficiently implemented on the bi-relational graph.

5 Discussion and Conclusion

We introduce the Bi-relational Graph (BG) model to associate proteins with multiple function classes simultaneously. By referring to the proteins and function classes equally as vertices, the order of matrix is reduced from nK to $n+K$. We consider a function class and its training proteins as a function group. In this way, the protein function prediction problem reduces to measuring how closely a specific protein is related to a given function group. Based on our previous work [8], we use network propagation to measure the relevance. During each iteration, the proteins receive label information from the function groups and their initial labels, updating labels from the previous iteration. Therefore, our algorithm can be understood as a generalization of the semi-supervised learning with local and global consistency [18] on a bi-relational graph. 5-fold cross validations on a yeast protein-protein interaction network compiled from the BioGRID database show that our algorithm can achieve superior performance compared to four state-of-the-art approaches.

As shown, the weight of the protein vertex in each function group has a very important influence on the prediction results. The performance of our algorithm is significantly improved even if a rather simple weighted strategy is introduced in our study. We expect that the performance can be further improved by adopting a more comprehensive weighting scheme. Here we fix the parameter $\alpha = 0.01$ as suggested by [18] and select the parameters β and γ through cross validation process. In fact, choosing these three parameters can be formulated as a combinatorial optimization problem, to which more attention should be paid. We leave all above-mentioned issues for our further investigation.

Acknowledgement. The author would like to thank Luqman Hodgkinson for a critical reading of the manuscript, and Lisa J. McQuay for her useful discussions and comments, and the anonymous reviewers for their constructive suggestions to improve the work.

References

1. Breitkreutz, B.J., Stark, C., Reguly, T., et al.: The BioGRID Interaction Database: 2008 update. Nucleic Acids Res. 36(Database issue), D637–D640 (2008)
2. Can, T., Camoğlu, O., Singh, A.K.: Analysis of proteinprotein interaction networks using random walks. In Proc. 5th International Workshop on Bioinformatics, pp. 61–68 (2005)

3. Chen, G., Song, Y., Wang, F., Zhang, C.: Semi-supervised Multi-label Learning by Solving a Sylvester Equation. In: SIAM International Conference on Data Mining (2008)
4. Erten, S., Bebek, G., Koyutürk, M.: Disease Gene Prioritization Based on Topological Similarity in Protein-Protein Interaction Networks. In: Bafna, V., Sahinalp, S.C. (eds.) RECOMB 2011. LNCS, vol. 6577, pp. 54–68. Springer, Heidelberg (2011)
5. Fan, R.-E., Lin, C.-J.: A Study on Threshold Selection for Multi-label Classification. Technical Report, National Taiwan University (2007)
6. Haveliwala, T.H.: Topic-sensitive pagerank. In: 11th International World Wide Web Conference (WWW), pp. 517–526 (2002)
7. Hishigaki, H., Nakai, K., Ono, T., Tanigami, A., Takagi, T.: Assessment of prediction accuracy of protein function from proteinCprotein interaction data. Yeast 18, 523–531 (2001)
8. Jiang, J.Q.: Multi-label Correlated Semi-supervised Learning for Protein Function Prediction. In: Chen, J., Wang, J., Zelikovsky, A. (eds.) ISBRA 2011. LNCS, vol. 6674, pp. 368–379. Springer, Heidelberg (2011)
9. Karaoz, U., Murali, T.M., Letovsky, S., Zheng, Y., Ding, C., Cantor, C.R., Kasif, S.: Whole-genome annotation by using evidence integration in functional-linkage networks. Proc. Natl. Acad. Sci. USA 101, 2888–2893 (2004)
10. Nabieva, E., Jim, K., Agarwal, A., Chazelle, B., Singh, M.: Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. Bioinformatics 21(suppl. 1), 302–310 (2005)
11. Pan, J.-Y., Yang, H.-J., Faloutsos, C., Duygulu, F.: Automatic Multimedia Cross-modal Correlation Discovery. In: The 10th ACM SIGKDD Conference, Seattle, WA, August 22-25 (2004)
12. Pavlidis, P., Weston, J., Cai, J., Grundy, W.N.: Gene functional classification from heterogeneous data. In: Proceedings of the Fifth Annual International Conference on Computational Biology. ACM Press, Montreal (2001)
13. Schwikowski, B., Uetz, P., Fields, S.: A network of proteinCprotein interactions in yeast. Nat. Biotechnol. 18, 1257–1261 (2000)
14. Vazquez, A., Flammini, A., Maritan, A., Vespignani, A.: Global protein function prediction from proteinCprotein interaction networks. Nat. Biotechnol. 21, 697–700 (2003)
15. Wang, H., Huang, H., Ding, C.: Image annotation using multi-label correlated Green's function. In: IEEE International Conference on Computer Vision (2009)
16. Wang, H., Huang, H., Ding, C.: Image Annotation Using Bi-Relational Graph of Images and Semantic Labels. In: IEEE International Conference on Computer Vision and Pattern Recognition (2001)
17. Zha, Z., Mei, T., Wang, J., Wang, Z., Hua, X.: Graph-based semi-supervised learning with multi-label. In: IEEE International Conference on Multiamedia and Expo (2008)
18. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Scholkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems (NIPS), vol. 16, pp. 321–328. MIT Press, Cambridge (2004)
19. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison (2005)

Graph-Based Decomposition of Biochemical Reaction Networks into Monotone Subsystems

Hans-Michael Kaltenbach[1], Simona Constantinescu[2], Justin Feigelman[2], and Jörg Stelling[1,2]

[1] Dep. Biosystems Science and Engineering, ETH Zurich, CH-4058 Basel
[2] Dep. Computer Science, ETH Zurich, CH-8092 Zurich
{hans-michael.kaltenbach,joerg.stelling}@bsse.ethz.ch

Abstract. Large-scale model development for biochemical reaction networks of living cells is currently possible through qualitative model classes such as graphs, Boolean logic, or Petri nets. However, when it is important to understand quantitative dynamic features of a system, uncertainty about the networks often limits large-scale model development. Recent results, especially from monotone systems theory, suggest that structural network constraints can allow consistent system decompositions, and thus modular solutions to the scaling problem. Here, we propose an algorithm for the decomposition of large networks into monotone subsystems, which is a computationally hard problem. In contrast to prior methods, it employs graph mapping and iterative, randomized refinement of modules to approximate a globally optimal decomposition with homogeneous modules and minimal interfaces between them. Application to a medium-scale model for signaling pathways in yeast demonstrates that our algorithm yields efficient and biologically interpretable modularizations; both aspects are critical for extending the scope of (quantitative) cellular network analysis.

1 Introduction

Advances in experimental technologies have enabled the large-scale analysis of cellular networks, for instance, those involved in gene regulation. In terms of understanding network function through formal modeling, however, the combination of network complexity and network dynamics poses substantial challenges. In particular, discrete models (e.g., based on graph theory, Boolean logic, or Petri nets) are scalable, but often quantitative aspects of systems dynamics are important. For such cases, much less scalable quantitative models, for instance in the form of ordinary differential equation (ODE) systems, need to be developed [12]. Examples for recently established large-scale dynamic models of signaling networks are an integrated model of yeast MAP kinase signaling [19] with 52 state variables or the epidermal growth-factor (EGF) network [4] with nearly 500 state variables. It is one of the great challenges of computational systems biology to develop tools that allow to analyze such complex models.

One strategy to cope with large networks is an analysis on the level of subsystems. This approach implies that a network is decomposed into smaller subsystems according to given criteria such as minimal overall complexity. Then, the subsystems and their interconnections are analyzed. While many examples exist for topology-based decompositions, a meaningful decomposition of dynamic network models necessarily needs to take the dynamics of the subsystems into account [1]. Examples of such strategies are the minimization of *retroactivity* [15], that is, the effect of a downstream component on a subsystem, modular response analysis [13], or analysis by stochastic independence of signals [3]. However, such strategies often require detailed system models initially. While stoichiometries of reactions are usually well-known, most parameters such as kinetic rate constants and initial conditions are often ill-defined. This poses particular problems for a network decomposition that takes into account the quantitative dynamics.

Recent theoretical results suggest that nontrivial conclusions about the network dynamics can be drawn by exploiting the constraints on the equation system imposed by the underlying network. For example, certain properties of graph [5,6] and matrix [2] representations of the network structure allow one to distinguish systems capable and incapable of multiple equilibria, respectively. Here, we are interested in applying the theory of monotone systems [16,8], extended to systems with inputs and outputs [17], which even allows to infer general results on the behavior of a system under perturbations. Importantly, these theories do not require knowledge of numerical parameter values, but only a specification of the stoichiometry of the reaction system and some weak, natural assumptions on the reaction rate laws (detailed below). In addition, important special cases of monotonicity can be established using a graph representation of the system, which transforms the problem from a continuous, nonlinear ODE system into a discrete graph problem.

We propose a new strategy to decompose a given reaction network into interconnected input-output monotone subsystems, each of which has very well-behaved responses to perturbations. The strategy comprises an initial decomposition using a greedy-approach that exploits an approximate solution of a MAX-CUT problem, followed by a refinement of the decomposition to minimize the interconnections of the subsystems. The algorithms are graph-based and rely on optimally cutting edges such that no subsystem contains a negative cycle. A decomposition is independent of parameter values and the exact form of rate laws, which enables the analysis of a system even with few data.

2 Monotone Systems

2.1 Dynamic Network Models

Consider a (bio)chemical reaction network with n chemical species $\mathcal{S} = \{S_1, \ldots, S_n\}$ and r reactions $\mathcal{R} = \{R_1, \ldots, R_r\}$. An irreversible reaction R_j is given by

$$R_j : \sum_{i=1}^{n} a_{j,i} S_i \rightarrow \sum_{i=1}^{n} b_{j,i} S_i,$$

where $a_{j,i}, b_{j,i} \in \mathbb{Z}$ are the *substrate* and *product molecularities*, respectively. The terms $N_{i,j} := b_{j,i} - a_{j,i}$ are called the *stoichiometries*. They describe the net production of species by the reaction and form the *stoichiometric matrix* $N := (N_{i,j})_{1\leq i\leq n, 1\leq j\leq r}$. We can always split a reversible reaction into two irreversible reactions, and thus always assume reactions to be irreversible. To capture the dynamics of the system, let $x_i(t) \geq 0$ denote the concentration of species S_i at time $t \geq 0$. The *state* of the system is then given by the concentration vector $x = (x_1(t), \ldots, x_n(t))^T$; as usual, the explicit dependence on time is suppressed.

Each reaction R_j is assigned a *rate equation* $v_j(x) : \mathbb{R}^n_{\geq 0} \to \mathbb{R}$, describing the velocity of the reaction. Here, we restrict the feasible rate equations to the class N1C. Informally, for irreversible reactions considered here, this ensures that an increase in a species concentration cannot decrease the rate of any reaction and that a species not participating in a reaction does not influence its rate. Formally, a network is N1C if

$$\forall x \in \mathbb{R}^n_{\geq 0} \forall i, j : N_{i,j} \cdot \frac{\partial v_j(x)}{\partial x_i} \leq 0 \text{ and } N_{i,j} = 0 \implies \frac{\partial v_j(x)}{\partial x_i} = 0.$$

Importantly, the stoichiometric matrix completely defines the topology of an N1C network. Note that this class is very general and it contains, for example, the mass-action, Michaelis-Menten, and Hill-type kinetic rate laws. For irreversible reactions, N1C also implies $\partial v_j(x)/\partial x_i \geq 0$.

With $v(x) = (v_1(x), \ldots, v_r(x))^T$, the overall temporal dynamics of the system is given by the set of n nonlinear ODEs

$$\frac{dx}{dt} = f(x) = N \cdot v(x).$$

2.2 Directed Species-Reaction Graph

Here, we are interested in a representation of the sign-structure of the system's Jacobian matrix $J(x) = (\partial f/\partial x)(x)$. It can be derived from the *directed species-reaction graph* G_{SR}, a directed, bipartite graph with signed edges, see Fig. 1A for an example. The graph's vertex sets are the species and reactions of the network, that is, $(\mathcal{S}, \mathcal{R})$. Specifically, if species S_i is a substrate of reaction R_j and thus $N_{i,j} < 0$, a positive edge (with label +) is drawn from the species to the reaction vertex. This reflects the positive (more precisely: nonnegative) influence that an increase in the species concentration has on the reaction rate. Similarly, a negative edge is added from the reaction back to the species to reflect the faster decrease of the species' concentration with increasing reaction rate. If the species is a product of the reaction, a positive edge is drawn from the reaction to the species. The restriction to N1C rate laws ensures that edges always have a well-defined sign and that there is at most one edge per direction between any two vertices.

The system can be extended with m inputs $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ and p outputs y given by $h : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^p$, and then takes the form

$$\frac{dx}{dt} = f(x, u) = N \cdot v(x, u) \quad \text{and} \quad y = h(x). \tag{1}$$

2.3 Identification of Monotonicity

A system (1) is called *monotone* if there are three partial orders, all denoted by $\prec$, on suitable subsets of $\mathbb{R}^m$, $\mathbb{R}^n$, and $\mathbb{R}^p$, respectively, such that for all initial conditions x_1, x_2, for all inputs $u_1(t), u_2(t)$ and for all times $t \geq 0$, we have that

$$x_1 \prec x_2 \text{ and } u_1(t) \prec u_2(t) \implies \phi(t; x_1, u_1(t)) \prec \phi(t; x_2, u_2(t)),$$

where $\phi(t; x_i, u_i(t))$ denotes the solution of the ODE system at time t, with initial condition x_i and (time-varying) input $u_i(t)$. Monotone systems exhibit several properties that make them very appealing for analyzing models in systems biology. Most importantly, they admit a single, global asymptotically stable steady state under mild additional conditions, and they respond "well-behaved" to perturbations. More details on monotone systems and their properties are given in [17].

Here, we are only concerned with orders induced by an *orthant cone* of the form $\mathcal{K} = \{x \in \mathbb{R}^n | e^T \cdot x \geq 0\}$, where $e \in \{-1, 1\}^n$. A partial order is then given by $x \prec y \iff y - x \in \mathcal{K}$; different orthant cones may be chosen for the three orders.

Establishing monotonicity of a system with respect to arbitrary cones is very difficult. In contrast, monotonicity with respect to orthant cones can be established by investigating the cycle structure of the *influence* or *interaction graph* of the system (see Fig. 1B and C for examples). This graph is a signed directed graph $G_{\text{I}} = (V, E)$ with one vertex for each species, input, and output, respectively. An edge is drawn if the corresponding partial derivative $\partial f_i / \partial x_j$, $\partial f_i / \partial u_j$, or $\partial h_i / \partial x_j$ (i.e., the corresponding entry in the Jacobian of the system) does not vanish identically for all x, u, t.

Assuming that the derivatives do not change sign, each edge $e \in E$ can be assigned the sign (+/-) of the derivative, leading to a signed graph. We can easily cover the case of changing signs by introducing a new vertex and splitting the corresponding edge (see example below) because this procedure does not change the structure of negative cycles in the graph. The influence graph thus describes how species influence each other. Its adjacency matrix is associated to the Jacobian matrix of the system, with a +1/-1 entry whenever the sign of the corresponding entry in the Jacobian is positive/negative for at least one state x, input u, and time t. Inputs and outputs are treated equivalently to species in the influence graph, the only difference being the computation of the edge sign. We can derive the influence graph and its edge signs by investigating all two-paths in the species-reaction graph from one species to another. Paths from a species to a reaction and back to the same species cause self-loops in the influence graph, which can be ignored in the further analysis (see [17] for an explanation).

For computational reasons, one often works with an undirected version of the influence graph. Clearly, having a negative cycle in the undirected version is a necessary condition for a negative cycle in the directed graph. Hence, a system is monotone with respect to some orthant cone, if the undirected influence graph does not contain a negative cycle [17].

2.4 Example Network

To illustrate the principles of monotone systems analysis, consider the reaction network $R_1 : A + B \rightarrow C$ and $R_2 : B + C \rightarrow A$ with three species, two reactions, and no inputs or outputs. Its dynamics is described by

$$\frac{d}{dt}\begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} v_1(A,B,C) \\ v_2(A,B,C) \end{pmatrix}.$$

The Jacobian matrix and the Jacobian's sign-pattern are given by

$$J(x) = \begin{pmatrix} -\frac{\partial v_1}{\partial A} & -\frac{\partial v_1}{\partial B} + \frac{\partial v_2}{\partial B} & \frac{\partial v_2}{\partial C} \\ -\frac{\partial v_1}{\partial A} & -\frac{\partial v_1}{\partial B} - \frac{\partial v_2}{\partial B} & -\frac{\partial v_2}{\partial C} \\ \frac{\partial v_1}{\partial A} & \frac{\partial v_1}{\partial B} - \frac{\partial v_2}{\partial B} & -\frac{\partial v_2}{\partial C} \end{pmatrix}(x), \quad \mathrm{sign}(J) = \begin{pmatrix} - & \mp & + \\ - & - & - \\ + & \pm & - \end{pmatrix}.$$

Three graphical representations are shown in Fig. 1: The directed, signed bipartite species-reaction graph (A) as well as the species-species influence graph with undefined edges (B) and with additional vertices to remove edges with undefined sign (C).

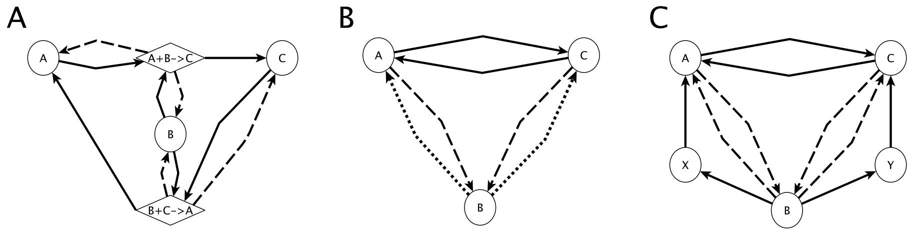


Fig. 1. Example network. **A**: directed species-reaction graph G_{SR}. **B**: influence graph G_{I} with undefined edge signs. **C**: influence graph for example network with additional vertices X and Y to split edges with undefined sign. Solid lines: positive, dashed lines: negative, dotted lines: undefined edges. Circles: species, diamonds: reaction.

This simple example illustrates that most biological networks are in fact not monotone. It suffices to have two species occurring on the same side in one, and on different sides in another reaction, to create a negative loop that makes the system non-monotone. To obtain monotonicity, in [17,7,10,18,9], methods were proposed to essentially "pull out" a minimal set of edges such that the resulting system does not contain a negative cycle and is therefore a monotone system. For *near-monotone* systems with few such edges, this strategy allows an analysis of the overall system. In contrast, our goal is to decompose a system into several, more homogeneous monotone subsystems that allow a coarse-grained analysis of the feedback structures and potentially provides a biological interpretation.

3 Algorithm for Monotone Systems Decomposition

3.1 Overview

Our aim is to compute a decomposition $\mathcal{M} = (M_1, \ldots, M_q)$ of a given reaction network into subsystems such that each subsystem M_i is connected and monotone. Note that different decompositions may differ in size, and thus q is part of the output of the method. A subsystem M_i is then identified with the subgraph induced by the corresponding vertices in either G_{SR} or, by projection on the species, G_{I}.

For evaluating a decomposition into monotone subsystems, we introduce the *interface* of each subsystem as the set of species concentrations and fluxes that need to be exchanged with other subsystems. A decomposition with smaller interfaces is then preferred to a decomposition with larger interfaces. Let $\mathcal{E}(R) \subseteq \mathcal{S}$ and $\mathcal{P}(R) \subseteq \mathcal{S}$ be the sets of educt and product species of a reaction $R \in \mathcal{R}$, respectively, and define $\mathcal{E}(\mathcal{X}) = \bigcup_{R \in \mathcal{X}} \mathcal{E}(R)$ for any $\mathcal{X} \subseteq \mathcal{R}$. Let $\mathcal{S}_M$ and $\mathcal{R}_M$ be the species and reactions of any subsystem M. Then, the interface of M with the remaining system is

$$\begin{aligned}\mathcal{I}(M) = &\{R \in \mathcal{R}_M \mid \mathcal{E}(R) \cup \mathcal{P}(R) \nsubseteq \mathcal{S}_M\}\\ &\cup\{R \notin \mathcal{R}_M \mid (\mathcal{E}(R) \cup \mathcal{P}(R)) \cap \mathcal{S}_M \neq \emptyset\}\\ &\cup\{S \in \mathcal{S}_M \mid S \in \mathcal{E}(\mathcal{R}\backslash\mathcal{R}_M)\}\\ &\cup\{S \notin \mathcal{S}_M \mid S \in \mathcal{E}(\mathcal{R}_M)\}.\end{aligned}$$

The first two sets define the incoming and outgoing fluxes and the second two sets define the incoming and outgoing species concentrations, respectively. A decomposition $\mathcal{M}$ is *optimal* if $\sum_{M \in \mathcal{M}} |\mathcal{I}(M)|$ is minimal. Importantly, the interface size is *not* the number of edges with one vertex in and one vertex outside the subsystem, as for example a reaction in $\mathcal{R}_M$ might have two product species in another subsystem M', but still only one flux needs to be exchanged.

As shown in [7,9,17], even finding a minimal set of edges to be removed to make the system monotone is hard and can be reduced to the MAX-CUT problem. In contrast to these publications, we are not interested in detecting the largest monotone subgraph, but in a decomposition of a reaction network into several interconnected monotone subsystems. For this, we use a three-phase strategy: In a first phase, we compute a decomposition that assigns each species to a subsystem and guarantees that each subsystem is monotone by avoiding negative cycles. In the second phase, the decomposition is translated to the decomposition $\mathcal{M}^0$ on the bipartite species-reaction graph to capture the interfaces. The third —and computationally most demanding— phase iteratively refines the initial decomposition by merging two subsystems and by splitting them again into monotone subsystems with minimal pair-wise interface. This procedure is repeated with randomly chosen pairs of subsystems until convergence, or until a predefined time or step number threshold is reached. In practice, this third phase often leads to merging of two subsystems into one, thereby reducing the total number of subsystems.

3.2 Phase 1: Initial Decomposition

We start with the trivial decomposition $\mathcal{M} = (\{S_1\}, \ldots, \{S_n\})$ of the influence graph G_{I}, such that each species is in its own (monotone) subsystem. Subsystems are then iteratively merged into larger monotone subsystems by joining their vertex sets, until merging is no longer feasible.

To determine which merges potentially lead to a non-monotone subsystem, we use the approximation algorithm for the modified MAX-CUT problem proposed in [7]. This algorithm computes a minimal set $I \subseteq E$ of inconsistent edges in the *undirected* signed influence graph G_{I} such that by removing these edges, the system becomes monotone. Merging is then performed such that no two vertices incident to an inconsistent edge are in the same subsystem. For this, we construct a $|\mathcal{S}| \times |\mathcal{S}|$ feasibility matrix F such that

$$F_{i,j} = \begin{cases} 0, & \text{if } \{i,j\} \notin E \\ -1, & \text{if } \{i,j\} \in I \\ +1, & \text{else.} \end{cases}$$

A positive entry $F_{i,j}$ indicates that the vertices of subsystems i, j can be merged and the induced subgraph is still monotone and connected; i, j are then called a *consistent pair.*

Two subsystems i, j are chosen uniformly at random from the set of all consistent pairs and merged into a new subsystem. The feasibility matrix is updated by

$$F_{i,k}, F_{k,i} \leftarrow \begin{cases} -1, & \text{if } F_{i,k} = -1 \text{ or } F_{j,k} = -1 \\ 0, & \text{if } F_{i,k} = 0 \text{ and } F_{j,k} = 0 \\ +1, & \text{else} \end{cases}$$

for all $k = 1, \ldots, n$ and $F_{k,j}, F_{j,k} \leftarrow 0$ for all $k \neq i$. In essence, this removes vertex j from the procedure and assigns the newly merged subsystem to vertex i. In each step, the number of consistent pairs decreases or stays constant. The merging of subsystems is continued until there are no more consistent pairs. The result is the initial decomposition $\mathcal{M}^0$ for all species.

3.3 Phase 2: Assignment of Reaction Vertices

In the second phase, we extend the subsystem assignment from G_{I} to G_{SR} by computing an initial assignment of reaction vertices. Because both flux and concentration need to be exchanged if a reaction and one of its substrates are in different subsystems, but only the flux for a product, we place the reaction rate into the largest subsystem that contains most of its substrates to ensure minimal subsystem interfaces for the given assignment of species. This yields the full specification of $\mathcal{M}^0$.

3.4 Phase 3: Refinement of Decomposition

The initial greedy strategy (phase 1) crucially depends on the set of inconsistent edges; the algorithm basically selects one edge per negative cycle, and it is not intended to create any useful subsystem decomposition. Thus, the initial decomposition $\mathcal{M}^0$ often contains many small subsystems that cannot be merged into larger ones anymore. However, they could disappear into a larger subsystem if assignments were "shuffled" slightly. Obviously, such a reassignment needs to ensure that all subsystems in the resulting decomposition are still monotone. However, while the interfaces change locally by adding or removing vertices, splitting all negative cycles requires a more global view on the subgraphs. Cycles might be partially overlapping and each cycle can be split in various positions which might lead to better or worse interfaces.

Correspondingly, the third phase employs a heuristic strategy by iteratively merging pairs of subsystems–potentially creating a non-monotone subsystem–and reassigning its vertices from the current decomposition $\mathcal{M}^i$ in iteration i to a new pair of subsystems with minimal interface, resulting in a new decomposition $\mathcal{M}^{i+1}$.

Let $M, M' \in \mathcal{M}^i$ be the two selected subsystems. We then determine all fundamental negative cycles of the subgraph $G_{\mathrm{M}\cup\mathrm{M}'} \subseteq G_{\mathrm{I}}$, induced by the two vertex sets of M, M', using a version of Johnson's algorithm for cycle enumeration [11] adapted to undirected graphs. If $G_{\mathrm{M}\cup\mathrm{M}'}$ does not contain a negative cycle, it is already monotone by itself. Otherwise, the graph $G_{\mathrm{M}\cup\mathrm{M}'}$ is re-partitioned into two monotone subsystems, potentially leading again to M and M'. In our experience, monotone subsystems typically contain much less than 30 vertices and their sizes are largely independent of the overall system size, making a cycle enumeration feasible.

For computing all feasible bi-partitions of the merged system into connected components, we employ a depth-first search strategy that starts from a vertex $v \in M \cup M'$, inducing two subsystems $M_1 = \{v\}, M_2 = (M \cup M')\backslash\{v\}$. The first subsystem is then iteratively grown by adding neighboring vertices from the second subsystem. Once a negative cycle appears in the first subsystem, we abort the depth-first search, backtrack, and resume at the last feasible solution because the cycle cannot vanish by adding more vertices. Thus, all subsequent solutions along this search path are necessarily infeasible. If neither subsystem contains a negative cycle, the interface is computed by assigning the reaction vertices as in Phase 2, and the solution is kept if the interface size is smaller than the previous minimum.

4 Application Example

To test our approach of monotone subsystems decomposition of a biochemical reaction network, we used a dynamic model that describes four mitogen-activated protein kinase (MAPK) pathways in *S. cerevisiae*. This model was published and described in [19] and we downloaded it as an SBML file from the BioModels database [14] (uncurated track). The network encompasses 52 chemical species and 58 reactions, including several MAPK cascades and a highly connected subnetwork of protein complex formation. We introduced minor modifications to the model to correct some obvious errors, in particular, by setting the reversibility of reactions according to [19].

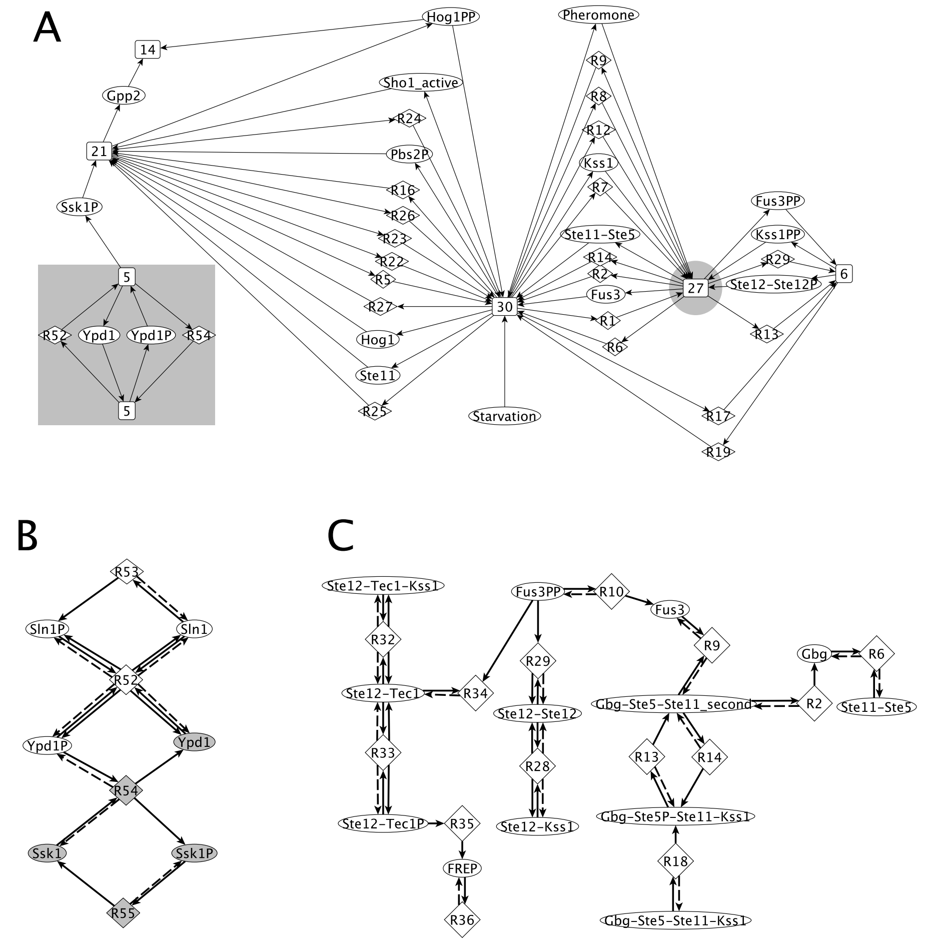


Fig. 2. Decomposition of the MAPK signaling network into monotone subsystems. **A**: Decomposition after 100 iterations of pairwise interface refinements. The number of subsystems is 8, the interface size 38. Rectangles denote subsystems, labeled with the number of vertices (species and reactions) assigned to them. The interfaces are given as ellipses for species concentrations and diamonds for reaction rates, labeled with their respective names. **B**: Interaction of two subsystems (given by grey box in **A**) with 5 vertices each. **C**: Subsystem with 27 vertices (given by grey circle in **A**) from the decomposition.

The initial decomposition using the greedy-approach and the reaction assignment yielded 10 subsystems and an interface size of 49, with 20 species concentrations and 29 reaction rates. The individual subsystems have sizes (31,21,11 (3x),8,7,6,2,1), given by their number of species and reaction vertices. Certain inputs (pheromone, starvation, and osmotic stress) are discarded if they are not assigned into a module with at least one other vertex, so the sum of subsystem sizes is smaller than the original number of vertices in the graph.

The result of the refinement after phase 3 (limited to 100 iterations) is shown in Fig. 2A. The algorithm converged after about two dozen iterations, and interface sizes and subsystems then stayed constant. The number of subsystems is eight. Hence, two pairs of subsystems were merged in the process, reducing the complexity of the representation. The subsystems now have individual sizes (30,27,21,14,6,5). Overall, the subsystems are therefore both larger and more homogeneous. Importantly, also the interfaces were reduced considerably, resulting in a total interface size of 38, with 16 species concentrations and 22 reaction rates being exchanged between subsystems. In addition, there is only one species concentration that needs to be exchanged between three subsystems, while all other interface components are pairwise. In contrast, the initial decomposition showed several interface components for more than two subsystems. Overall, thus, with respect to the aim of obtaining a globally minimal (in the sense defined above) system decomposition, the iterative refinement using a randomized algorithm proved critical.

While not intended by the algorithm, the decomposition is also interpretable in more biological terms. In particular, with each subsystem being monotone, negative feedback regulation must yield at least two subsystems. An example is shown in Fig. 2B, which depicts the two subsystems marked by the grey rectangle in Fig. 2A. These subsystems are a decomposition of a typical phosphorylation cascade that is inherently non-monotone. Large subsystems with many species and reactions also often contain parts of the network with little regulation. An example is given in Fig. 2C, which shows the internal subsystem centered around the scaffold protein Ste11 that is given by the grey circle in Fig. 2A. Multiple edges between vertices as well as seemingly negative cycles are due to the reversibility of the original reactions, which is restored by the algorithm after the reaction was split into two irreversible reactions for the analysis. Overall, the size and interconnection of the subsystems give a first impression of the complexity of interactions in the various parts of the network model.

5 Conclusion

We presented a new strategy for decomposing large-scale dynamic models of biochemical reaction networks into interconnected subsystems. It employs results from monotone systems theory and aims at finding an optimal decomposition of a system into monotone components, which immediately guarantees various important dynamic properties for these components. In particular, a monotone system exhibits “well-behaved” responses to perturbations.

By restricting ourselves to the case of systems that are monotone with respect to so-called orthant cones, the problem of determining a system's monotonicity can be translated into analyzing a bipartite graph and an associated influence graph closely related to the sign-structure of the Jacobian matrix of the system. The decomposition itself is computed using a three-phase graph-based algorithm that first computes a valid decomposition of the chemical species into monotone subsystems, but not necessarily with minimal interfaces. After optimally assigning the reaction rates to the subsystems, a third phase iteratively refines the decomposition by merging and re-partitioning randomly selected pairs of subsystems, while keeping all subsystems monotone.

Many improvements can be imagined for the suggested algorithms. It would be of particular interest to avoid the influence graph altogether and to perform a decomposition directly on the directed species-reaction graph. In addition, devising an algorithm that more globally (approximately) optimally cuts all fundamental negative cycles and gives minimal interfaces could give an improvement of the decomposition. A first step could be to merge more than two subsystems for the iterative improvement.

Despite these limitations, our method gave meaningful results for a medium-size dynamic model of integrated yeast MAP-kinase pathways. While the initial decomposition is already useful, the refinement step clearly improves the decomposition significantly by simultaneously reducing both the number of subsystems, and the interface sizes.

Overall, we emphasize that the subsystems are not defined by topological criteria such as connectedness, but rather by certain desired dynamic properties. Nevertheless, they can be identified without parameter values and with only minimal knowledge about reaction rates. This is a typical setting in systems biology, which makes the approach very appealing to models in this field. In future applications, for instance, the guaranteed dynamic features can further be exploited to give a coarse-grained analysis of a system's feedback structure.

Acknowledgements. The authors would like to thank Markus Beat Duerr, Christian Mayer, and Mario Marchisio for helpful discussions. Financial support by the EU FP7 project UNICELLSYS is gratefully acknowledged.

References

1. Alexander, R.P., Kim, P.M., Emonet, T., Gerstein, M.B.: Understanding modularity in molecular networks requires dynamics. Sci. Signal. 2(81), pe44 (2009)
2. Banaji, M., Donnell, P., Baigent, S.: P matrix properties, injectivity, and stability in chemical reaction systems. SIAM J. Appl. Math. 67(6), 1523–1547 (2007)
3. Bowsher, C.G.: Information processing by biochemical networks: a dynamic approach. Journal of The Royal Society Interface 8, 186–200 (2010)
4. Chen, W.W., Schoeberl, B., Jasper, P.J., Niepel, M., Nielsen, U.B., Lauffenburger, D.A., Sorger, P.K.: Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. Mol. Syst. Biol. 5, 239 (2009)
5. Craciun, G., Feinberg, M.: Multiple equilibria in complex chemical reaction networks: I. The injectivity property. SIAM J. Appl. Math. 65(5), 1526–1546 (2005)

6. Craciun, G., Feinberg, M.: Multiple equilibria in complex chemical reaction networks: II. The species-reaction graph. SIAM J. Appl. Math. 66(4), 1321–1338 (2006)
7. DasGupta, B., Enciso, G.A., Sontag, E., Zhang, Y.: Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. Biosystems 90(1), 161–178 (2007)
8. Hirsch, M., Smith, H.: Monotone dynamical systems. In: Handbook of Differential Equations: Ordinary Differential Equations, vol. 2, pp. 239–357 (2006)
9. Hüffner, F., Betzler, N., Niedermeier, R.: Optimal edge deletions for signed graph balancing. In: Demetrescu, C. (ed.) WEA 2007. LNCS, vol. 4525, pp. 297–310. Springer, Heidelberg (2007)
10. Iacono, G., Ramezani, F., Soranzo, N., Altafini, C.: Determining the distance to monotonicity of a biological network: a graph-theoretical approach. IET Systems Biology 4(3), 223–235 (2010)
11. Johnson, D.B.: Finding all the elementary circuits of a directed graph. SIAM J. Comput. 4(1), 77–84 (1975)
12. Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. Nat. Rev. Mol. Cell. Biol. 9(10), 770–780 (2008)
13. Kholodenko, B.N., Kiyatkin, A., Bruggeman, F.J., Sontag, E.D., Westerhoff, H.V., Hoek, J.B.: Untangling the wires: A strategy to trace functional interactions in signaling and gene networks. Proc. Natl. Acad. Sci. USA 99(20), 12841–12846 (2002)
14. Li, C., Donizelli, M., Rodriguez, N., Dharuri, H., Endler, L., Chelliah, V., Li, L., He, E., Henry, A., Stefan, M.I., Snoep, J.L., Hucka, M., Le Novère, N., Laibe, C.: BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models.. BMC Systems Biology 4, 92 (2010)
15. Saez-Rodriguez, J., Gayer, S., Ginkel, M., Gilles, E.D.: Automatic decomposition of kinetic models of signaling networks minimizing the retroactivity among modules. Bioinf. 24(16), i213–i219 (2008)
16. Smith, H.: Monotone dynamical systems: an introduction to the theory of competitive and cooperative systems. Mathematical Surveys and Monographs, vol. 41. American Mathematical Society AMS, Providence (1995)
17. Sontag, E.: Monotone and near-monotone biochemical networks. LNCIS, vol. 357, pp. 79–122 (2007)
18. Soranzo, N., Ramezani, F., Iacono, G., Altafini, C.: Graph-theoretical decompositions of large-scale biological networks. Automatica (2010), conditionally accepted
19. Zou, X., Peng, T., Pan, Z.: Modeling specificity in the yeast MAPK signaling networks. J. Theor. Biol. 250(1), 139–155 (2008)

Seed-Set Construction by Equi-entropy Partitioning for Efficient and Sensitive Short-Read Mapping

Kouichi Kimura[1], Asako Koike[1], and Kenta Nakai[2]

[1] Central Research Laboratory, Hitachi Ltd., 1-280 Higashi-Koigakubo, Kokubunji, Tokyo, 185-8601, Japan
[2] The Institute of Medical Science, The University of Tokyo, 4-6-1 Shirokane-dai, Minato-ku, Tokyo, 108-8639, Japan

Abstract. Spaced seeds have been shown to be superior to continuous seeds for efficient and sensitive homology search based on the seed-and-extend paradigm. Much the same is true in genome mapping of high-throughput short-read data. However, a highly sensitive search with multiple spaced patterns often requires the use of a great amount of index data. We propose a novel seed-set construction method for efficient and sensitive genome mapping of short reads with relatively high error rates, which uses only continuous seeds of variable length allowing a few errors. The seed lengths and allowable error positions are optimized on the basis of entropy, which is a measure of ambiguity or repetitiveness of mapping positions. These seeds can be searched efficiently using the Burrows-Wheeler transform of the reference genome. Evaluation using actual biological SOLiD sequence data demonstrated that our method was competitive in speed and sensitivity using much less memory and disk space in comparison to spaced-seed methods.

1 Introduction

Since the advent of new-generation DNA sequencers, sequencing throughputs have greatly increased and costs have drastically fallen. This has posed many challenging problems in sequence analysis, in particular, genome mapping of unprecedented amounts of sequencing data [7,19]. Thus, many mapping tools have been developed that are designed for vast amounts of short reads, typically tens of millions of short reads with lengths from 30 to 75. Among them, ELAND (Cox, unpublished) and MAQ [12], early popular tools designed for Illumina Genome Analyzer (GA), use *gapped seeds* for sensitive search. Recently introduced tools based on the Burrows-Wheeler transform, such as BWA [11], Bowtie [10] and SOAP2 [14], are now most widely used for mapping Illumina GA data because of their overwhelming speed. The Burrows-Wheeler transform is a memory-efficient alternative to a suffix array and widely used in text search and data compression problems [3,1].

In addition to speed, sensitivity is a major concern in genome mapping of short reads obtained from Applied Biosystems SOLiD sequencers. This is because a relatively high raw sequencing error rate should be assumed at first although high-quality data are eventually obtained by error correction based on the mapping results and due to the redundancy of the two-base encoding scheme [2]. BFAST [8], one of the most powerful tools for mapping SOLiD reads, uses *spaced seeds*, which have been used in many efficient and sensitive homology search tools [16,13,18]. BFAST is far more sensitive than BWA although it requires rather large amounts of memory and disk space (e.g., 17 GB of memory and 121 GB of disk space for index data in human genome mapping [8]).

In this article, we introduce a new seed-set construction method for efficient and sensitive genome mapping of short reads following the seed-and-extend paradigm. It uses only *continuous seeds* that can be explored efficiently with the Burrows-Wheeler transform. However, for higher sensitivity, it uses shorter seeds allowing a few errors per seed. The use of such seeds would result in prohibitively large computational costs unless they were carefully chosen. Thus, we select a suitable set of seeds for each read in a context-dependent manner, which is particularly important when we encounter with repetitive sequences. The concept underlying our method and some experimental evaluation results are presented.

2 Equi-entropy Partitioning Method

2.1 Soft Seeds

In the conventional seed-and-extend paradigm for string search problems, *seeds* mean short substrings of a query string that also appear only a limited number of times in a large target string. They are expected to quickly reduce the number of potential hit positions of the query in the target, which are later examined by *extensions* at more computational costs typically with the dynamic programming method. We refer to conventional seeds, which should exactly match substrings in the target string, as *hard seeds*.

In sensitive genome mapping of short reads at high error rates, the utility of hard seeds is limited due to the strong requirement for exact matching: there may be few if any intact seeds free from error. For example, a read of length 50 bp with three errors may not be detected by any hard seeds with a length of 13 bp or more. That is, with hard seeds, the sensitivity of mapping quickly degrades as the error rate increases. We thus introduce *soft seeds*, which allow a few errors inside. For clarity, we make the following definition.

Definition 1. *Let s be a subsequence of query read q, let k be a small non-negative integer, and let G be a target genome sequence. $M(k/s)$ denotes the set of all subsequences of G that can be matched with s with at most k errors (at most k base pairs of substitutions, insertions, or deletions in total).*

For $k = 0$, we refer to $M(0/s)$ as hard seed s; for $k > 0$, we refer to $M(k/s)$ as soft seed s with at most k errors. Note that $M(k/s)$ represents a set of matching positions (in the genome) of the seed, hence a set of *potential hit positions* of query q found by seed s with at most k errors. Thus, we will interchangeably use the term "seed" to mean: a sequence s with the allowable number of errors k, or the set of its matching positions in the genome.

2.2 Partitioning Read into Seeds

For sensitive and efficient search, the set of seeds should be carefully chosen. They should cover almost all of the query string with minimal overlap. Therefore, we partition a query read into a set of seeds. Although partially overlapped partitionings are acceptable in general, we mainly focus on disjoint partitionings.

When a read contains at most $n-1$ errors and is partitioned into n parts, at least one of the parts contains no errors. Thus, a mapping with at most $n-1$ errors can always be found by using one of these n hard seeds. More generally, for any non-negative integer k, when a read contains at most $(k+1)n-1$ errors and is partitioned into n parts, at least one of the parts contains at most k errors. Thus, a mapping with at most $(k+1)n-1$ errors can always be found by using one of these n soft seeds with at most k errors. We refer to $(k+1)n-1$ as *the tolerable number of errors*, which is a measure of the sensitivity of the seed set. For example, for $n=3$ and $k=1$, a mapping of a read of length 50bp with up to five errors can always be found by using a set of three soft seeds with lengths of 16, 17, and 17 bp each with at most one error.

Note that the tolerable number of errors is given in terms of n and k. Thus, we can optimize the seed lengths for efficient search without affecting the sensitivity as long as the same values of n and k are used. Larger values of n and k result in improved sensitivity at the expense of higher computational cost. We mostly use $k=1$ for efficiency, and $k=2$ only for special cases. Different values of n can be used for different reads, which is helpful in coping with repetitive reads.

2.3 Entropy and Adjustment of Seed Lengths

One can regard the role of a seed as an informant of potential hit positions for a query. Without seeds, hits can be anywhere in the genome, and the amount of ambiguity is represented as $H_G = \log_2 g$ bits in terms of information theory, where g denotes the reference genome size. When seed s restricts the potential hits to a limited number $\mathcal{W}(s)$ of positions, the amount of ambiguity is reduced to $\mathcal{H}(s) = \log_2 \mathcal{W}(s)$ bits. In other words, the amount of information seed s provides is given by $\mathcal{I}(s) = H_G - \mathcal{H}(s)$ bits. For example, if a seed has many hits in repetitive regions, $\mathcal{H}(s)$ takes a large value and $\mathcal{I}(s)$ takes a small value. We will use $\mathcal{H}(s)$ as a convenient measure of ambiguity or repetitiveness for seed s instead of $\mathcal{W}(s)$, the number of potential hits, which takes values with a wide range of magnitudes. We refer to $\mathcal{H}(s)$ as the *entropy* of seed s since it measures the amount of ambiguity of unresolved mapping positions in terms of Shannon entropy.

When a query read q is covered by a set of a fixed number (m) of disjoint seeds of variable lengths: $q = s_1 \sqcup \cdots \sqcup s_m$, the overall computational cost is largely determined by the total number of potential hits: $\mathcal{W} = \sum \mathcal{W}(s_i)$, since their extensions are the most computationally intensive. Thus, for efficiency, $\mathcal{W}$ should be minimized by adjusting the boundaries of seeds. Applying the variational principle, one observes that the number of potential hits for each seed should be almost equally small: $\mathcal{W}(s_i) \lesssim \bar{W}$ $(\forall i)$ for some constant $\bar{W} > 0$.

Thus, giving an appropriate constant $\bar{H}$, we may use a set of minimal-length seeds satisfying $\mathcal{H}(s) \leq \bar{H}$. As a result, seed lengths are adjusted in a context-dependent manner: longer seeds are selected for use in repetitive regions, and shorter seeds are selected for use in unique regions. Thus, we can avoid the enormous number of useless hits in repetitive regions that would be encountered if fixed-length seeds were used.

2.4 Computation of Entropy

For a hard seed s with $k = 0$, the number of exact matches $\mathcal{W}(s)$ and hence the entropy $\mathcal{H}(s) = \log_2 \mathcal{W}(s)$ can be computed very efficiently using the Burrows-Wheeler transform [3]. In fact, the computation is reduced to repeatedly computing *rank functions* on the transformed genomic sequence at most twice the seed-length number of times [5,15], and rank functions can be computed in a small constant time even for large genomes [4,6].

However, it is not an easy task to compute the entropy for soft seed s with $k > 0$ in general. All of the approximate matches within edit distance k should be enumerated. In terms of a suffix tree search, the number of branches that should be explored (N) increases rapidly as search depth d (up to seed length ℓ) increases. When N is less than the genome size (e.g., 3×10^9 for human) at the beginning of the search, N may increase as rapidly as the exponential in d^k, more precisely, as rapidly as 8^C with $C = {}_dC_k$, if eight types of errors — a substitution with one of three alternative bases, a deletion, or an insertion of one of four bases — are considered.

For clarity, let $\mathcal{H}^0(s)$ denote the entropy of a hard seed s, and let $\mathcal{H}^k(s)$ denote the entropy of a soft seed with at most $k > 0$ errors. Although $\mathcal{H}^0(s)$ and $\mathcal{H}^1(s)$ can take quite different values, we have experimentally observed that they are largely correlated, and the former can be used as a convenient approximate lower bound for the latter. Therefore, in adjusting the lengths of soft seeds with $k = 1$, we use entropy $\mathcal{H}^0(s)$ instead of $\mathcal{H}^1(s)$.

2.5 Bidirectional Search

When $k = 1$, the soft seed $M(1/s)$ can be computed efficiently by a bidirectional search (Figure 1). Let seed s be divided into two segments, s_0 and s_1. If an approximate matching of s has a single error, the error should be located either in the first segment s_0 or in the second segment s_1. In the former case, a desired matching is given by a combination of an exact matching on s_1 and an approximate matching with an error on s_0. For efficient search, one should examine them from right to left:

1. search for exact matches on s_1,
2. extend them into s_0 with at most one error.

The exact matching on s_1 reduces the entropy (ambiguity) to $\mathcal{H}(s_1)$ in the first step, and imposes constraints on the possible errors in the extended part (s_0) in the second step. Likewise, in the latter case, a desired matching is given by a

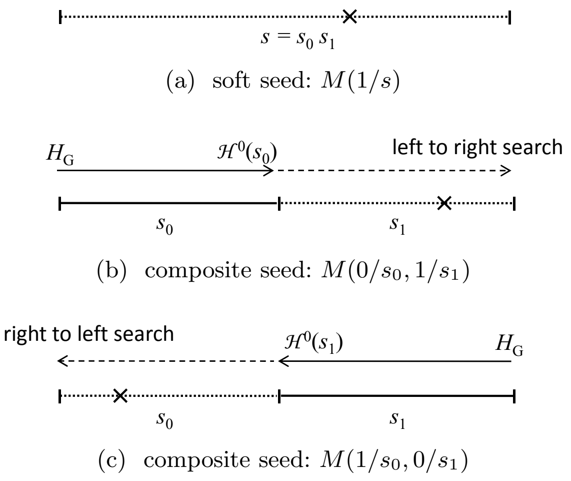


Fig. 1. Bidirectional search of approximate matching of seed with a single error

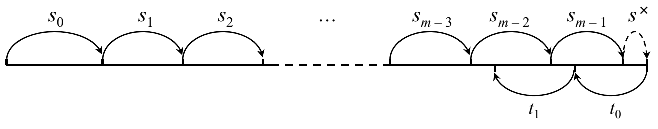


Fig. 2. Partitioning of read into equi-entropy segments

combination of an exact matching on s_0 and an approximate matching with an error on s_1, which should be examined from left to right. For convenience, we introduce notations for each of these concatenated matchings on s.

Definition 2. *Let s be a subsequence of query read q, which is divided into two non-empty segments, s_0 and s_1 with $s = s_0 s_1$. Let k_0 and k_1 be small non-negative integers and G be the target genome sequence. $M(k_0/s_0, k_1/s_1)$ denotes the set of all subsequences of G that can be matched with s with at most k_0 errors on s_0 and at most k_1 errors on s_1.*

We refer to $M(k_0/s_0, k_1/s_1)$ as *a composite seed.* Thus, a soft seed with at most one error can be obtained by a union of two composite seeds:

$$M(1/s) = M(0/s_0, 1/s_1) \cup M(1/s_0, 0/s_1),$$

and can be computed efficiently by the the bidirectional search provided that the both entropies, $\mathcal{H}^0(s_0)$ and $\mathcal{H}^0(s_1)$, are small.

2.6 Equi-entropy Partitioning

According to the above discussion, we may first partition a read into segments of almost equally low entropies and then make concatenated pairs in order to get a covering of read by soft seeds with $k = 1$. More precisely, we partition a read from left to right into segments $s_0, s_1, \ldots, s_{m-1}$ and possibly with a remainder $(s^\times)$ (*equi-entropy partitioning*, Figure 2). Each s_i is chosen so that it satisfies $\mathcal{H}(s_i) \le H$ and has the minimal length, where H is a chosen constant referred to as the *entropy parameter*. The remainder should satisfy $\mathcal{H}(s^\times) > H$ unless it is empty. The number of segments, m, referred to as *partition number*, may differ from read to read even if the read lengths are the same. Seeds are then given by pairing them: $s_0s_1, s_2s_3, s_4s_5, \ldots$. We do not use seeds like $s_1s_2, s_3s_4, \ldots$ in order to avoid overlapping searches. However, if the partition number is odd, the last segment, s_{m-1}, is lost in these seeds. Accordingly, we repartition the right end of the read when the partition number turns out to be odd. We take out two additional segments, t_0 and t_1, from the right end of the read to the left so that they satisfy $\mathcal{H}(t_i) \le H$ $(i = 0, 1)$ and have minimal lengths, as shown in Figure 2. Now, for simplicity, assume that the partition number is equal to three and consider the extended remainder (Figure 3):

$$\tilde{r} = s_0 \sqcup s_1 \sqcup s_2 \sqcup s^\times = s_0 \cup s_1 \cup s_2 \cup t_0 \cup t_1.$$

Let r_0 be the complement of $t_0 \cup t_1$ in $\tilde{r}$, r_2 be the complement of $s_0 \cup s_1$ in $\tilde{r}$, and r_1 be the complement of $r_0 \cup r_2$ in $\tilde{r}$, namely,

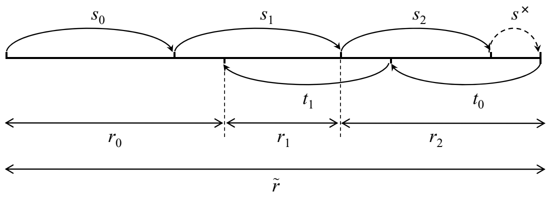
$$r_0 = \tilde{r} \setminus (t_0 \cup t_1), \quad r_2 = \tilde{r} \setminus (s_0 \cup s_1), \quad r_1 = \tilde{r} \setminus (r_0 \cup r_2).$$

Thus, the extended remainder, $\tilde{r}$, is repartitioned into these three segments: $\tilde{r} = r_0 \cup r_1 \cup r_2$. In general, when the partition number is odd and greater than three, r_0, r_1, and r_2 can be defined similarly by substituting $\{s_0, s_1, s_2\}$ with $\{s_{m-3}, s_{m-2}, s_{m-1}\}$.

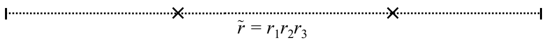
Now we define seed set $\mathcal{S}$:

$$\mathcal{S} = \begin{cases} \{M(1/s_0s_1), M(1/s_2s_3), \ldots, M(1/s_{m-2}s_{m-1})\} \\ \qquad\qquad\qquad\qquad\qquad\qquad (m : \text{even}), \\ \{M(1/s_0s_1), M(1/s_2s_3), \ldots, M(1/s_{m-5}s_{m-4}), \\ \qquad\qquad M(0/r_0, 2/r_1), M(1/r_1r_2)\} \quad (m \ge 3 : \text{odd}), \\ \{M(0/q)\} \qquad (m = 1), \end{cases}$$

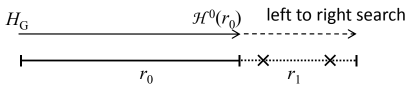
where q is the query read and $M(0/r_0, 2/r_1)$ represents a composite seed that allows no errors in r_0 and at most two errors in r_1. This composite seed can be examined efficiently from left to right since the exact matching in r_0 reduces the entropy to $\mathcal{H}^0(r_0) \le \mathcal{H}(s_0) \le H$. Soft seed $M(1/r_1r_2)$ can also be examined efficiently by using a bidirectional search because $r_1r_2 = t_1t_0$ and $\mathcal{H}(t_0), \mathcal{H}(t_1) \le H$. When m is even, remainder $s^\times$ is simply neglected in constructing $\mathcal{S}$, which turns out to be nevertheless useful, as shown in the following proposition. Strictly speaking, we should separately consider several degenerate cases in which some of the boundaries of s_i and t_j coincide with each other. We omit these details since they can be treated similarly.



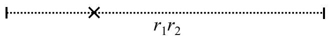
(a) repartitioning of extended remainder $\tilde{r}$



(b) soft seed: $M(2/\tilde{r})$



(c) composite seed: $M(0/r_0, 2/r_1)$



(d) soft seed: $M(1/r_1 r_2) = M(1/t_1 t_0)$

Fig. 3. Repartitioning of extended remainder ($m = 3$)

Proposition 1. *If partition number m is positive, the tolerable number of errors of seed set $\mathcal{S}$ is $m - 1$. That is, if the read has a mapping with at most $m - 1$ errors, it is always detected by at least one of the seeds in $\mathcal{S}$.*

Remark. Note that the number of seeds in $\mathcal{S}$ is about half the partition number; more precisely, we have $|\mathcal{S}| = \lfloor (m + 1)/2 \rfloor$.

Proof. The proof is straightforward and hence omitted. □

2.7 Adjusting Entropy Parameter *H*

The larger the entropy parameter (H), the larger the partition number (m) and the greater the sensitivity for erroneous reads. However, the computational cost also increases. In the trade off between sensitivity and cost, the entropy of seeds should take a reasonable value (say, five), corresponding to the ambiguity associated with a moderate number of possibilities (say, $32 = 2^5$). Since H controls the entropy of segments (halves of seeds) but not the seeds themselves, we consider the effect of H on the entropy of seeds.

Intuitively speaking, if the genome sequence is random, and if a seed (s) consists of two probabilistically independent adjacent segments, s_0 and s_1, the sum of their information amounts to the information of the seed:

$$\mathcal{I}(s) = \mathcal{I}(s_0) + \mathcal{I}(s_1).$$

Namely, in terms of entropy (ambiguity), we have

$$\mathcal{H}(s) = \mathcal{H}(s_0) + \mathcal{H}(s_1) - H_G.$$

Since s_0 and s_1 are chosen such that $\mathcal{H}(s_0) \sim H$ and $\mathcal{H}(s_1) \sim H$, we have

$$H \sim (H_G + \mathcal{H}(s))/2.$$

Therefore, reasonable values of H are somewhat larger than $H_G/2$, for example, around $(\log_2(3 \times 10^9) + 5)/2 \sim 18.2$ for the human genome (3 Gbp).

This intuitive discussion is actually not correct since genome sequences are not random and contain many repetitive sequences and since adjacent segments may be correlated. Nevertheless, the above discussion holds approximately for the majority of seeds, which we have experimentally observed. Thus, the H value selected above serves as a rough guide. However, when a seed is repetitive, the additivity of segment information does not hold, and $\mathcal{H}(s)$ may take a large value. In that case, the seed may have an enormous number of hit positions and prohibitively large computational costs will be required in their extensions. Therefore, we introduce a heuristic parameter, K: if $\mathcal{H}(s) > K$, we do not try to extend the alignment of that seed. K is referred to as the *pruning parameter*. Note that K is an upper bound on the entropy of seeds to be explored and that H is an upper bound on the entropy of segments (halves of seeds). Thus, we should choose a K that is sufficiently greater than $2H - H_G$, otherwise most seeds will be lost.

3 Experimental Results and Discussion

3.1 Implementation Issues

To evaluate the performance of the proposed method, we implemented a prototype genome mapping program in C++, referred to as *EEP*, for SOLiD color-space reads. Several implementation issues are briefly described here.

Nucleotide sequences of reference genomes were first converted into sequences of four colors (represented as 0, 1, 2, 3) in accordance with the SOLiD two-base encoding scheme [2]. The Burrows-Wheeler transform was applied to the sequences, and the results were formatted into a convenient data structure, *hierarchical binary strings* [9].

Query reads, given as sequences of colors, were compared with the genomic color sequences following the seed-and-extend paradigm. Seed sets were constructed using the equi-entropy method with appropriate entropy and pruning parameters, H and K. The hit positions of the seeds were efficiently calculated using rank functions on the Burrows-Wheeler transform of the reference genome [9]. Extensions were computed using a modified version of Myers' bit-parallel dynamic programming algorithm [17]. For the sake of bit-parallelism, unit costs for all substitution/insertion/deletion errors were assumed. Finally, optimal mapping positions on the reference genome with the minimal number of color-space edits (referred to as *edit distance*) were identified.

As for the human genome (3.2×10^9 Gbp), the amount of disk space used by the formatted index data was about 4.1 GB, and the amount of memory required for mapping was about 4.2 GB.

3.2 Performance Estimation Using Random Reads

We examined the performance of EEP using the reference human genome (UCSC hg19) and simulated read data: 50-bp-long subsequences randomly taken from the reference genome, converted into color sequences in accordance with the two-base encoding scheme [2], and introduced a fixed number (e) of random substitution errors.

Mapped ratio is defined as the number of reads mapped anywhere on the genome divided by the total number of reads. Figure 4 shows mapped ratio for different values of H and e without any pruning ($K = \infty$). For example, for $H = 18$ (closest to the 18.2 discussed in Section 2.7), almost all reads with five errors, about 90% of reads with six errors, and many of reads with more errors were mapped.

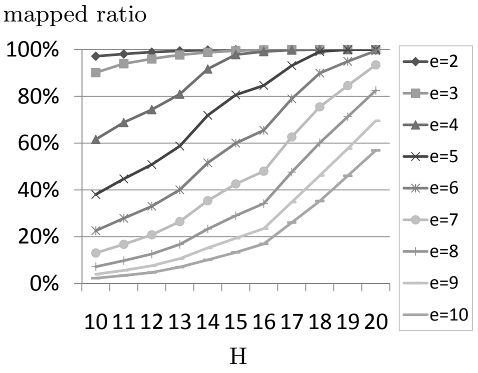


Fig. 4. Mapping ratio of EEP with different values of H for data with different number of errors per read ($e = 2, 3, \ldots, 10$)

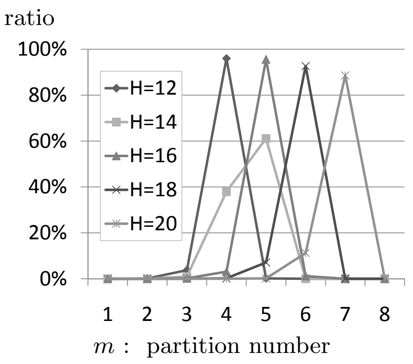


Fig. 5. Distributions of partition number m for different values of H for random reads with $e = 5$ (five errors each)

Table 1. Comparison of Partitioning Methods. A hundred thousand random reads of 50 bp with five substitution errors each ($e = 5$) are mapped on the human genome using exactly the same method except for the partitionings.

(a) Equi-Entropy Partitioning (EEP)

H	12	16	18	20
mapped ratio	50.9%	84.6%	99.1%	100.0%
time(sec.)†	109	495	2,383	15,670

(b) Equi-Length Partitioning (ELP)

m	4	5	6	7
mapped ratio	38.8%	80.2%	100.0%	100.0%
time(sec.)†	408	2,242	8,426	28,025

† with a single core of an Intel Xeon E5540 (2.53 GHz) and 16 GB of memory.

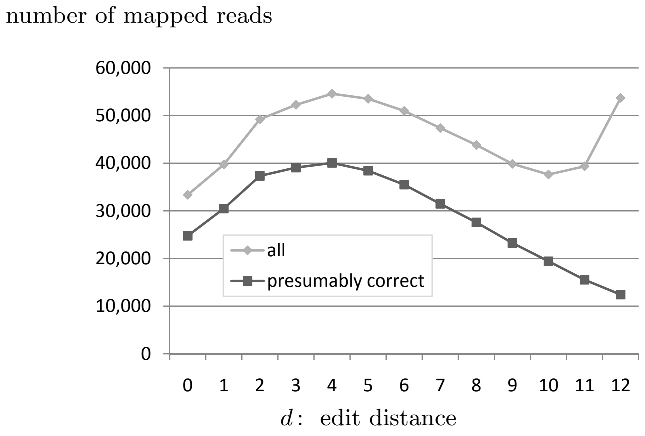


Fig. 6. Number of mapped reads for HLA-targeted sequence data

We compared the performances of the equi-entropy partitionings (EEP) with those of *equi-length partitionings* (ELP). As shown in Figure 5, choosing the entropy parameter $H = 12, 16, 18, 20$ mostly corresponds to choosing the partition number $m = 4, 5, 6, 7$. As shown in Table 1, EEP performed better than ELP in terms of both mapping sensitivity and speed.

3.3 Performance Estimation Using Actual Biological Data

We examined the performance of EEP on actual biological data[1], HLA (human leukocyte antigen)-targeted single-end SOLiD sequence data of length 50 bp. Although the correct mapping positions are unknown, the overall correctness of the mapping results can be evaluated by using the ratio of mappings that were successfully mapped to the HLA region, which occupies only about 0.1% of the genome. Thus, when a mapping position is in the HLA region, we say that it is "presumably correct". Note that there are two major reasons for presumably false mappings: (i) reads were in fact taken from outside the HLA region due to the limited sequence capture efficiency, and (ii) wrong mapping positions were selected (with the minimum number of mismatches) due to accidental sequencing errors or polymorphisms. We chose $H = 16$ and $K = 10$ to give higher priority to the mapping speed, and searched for mappings with up to 12 errors per read.

We examined how the ratio of presumably correct mappings decreases as the edit distance increases. A million reads of HLA-targeted sequence data were mapped to the reference human genome. Of them, 595,375 were mapped to somewhere in the genome, and 375,284 of which were presumably correct. As shown by the plots of these numbers by edit distance (Figure 6), presumably false mappings began to increase rapidly as the edit distance increased beyond

[1] We are grateful to Professor Ituro Inoue and Dr. Kazuyoshi Hosomichi of the Institute of Medical Sciences at Tokai University for providing us these sequence data.

Table 2. Comparison of performances on HLA-targeted data (10,000,000 reads)

program	EEP	BFAST	BWA
computation time[†]	12h1m48.9sec	16h20m10.5sec	4h11m40.6sec
	12h38m14.3sec	16h29m30.6sec	4h15m23.7sec
	12h14m6.0sec	16h27m40.8sec	4h31m59.6sec
	11h54m13.7sec	16h25m2.7sec	3h57m33.6sec
	11h37m57.8sec	16h35m4.7sec	4h12m41.0sec
number of all mapped reads	6,350,564	6,531,859	4,991,727
	6,092,972	6,359,180	4,578,039
	6,248,750	6,455,366	4,830,035
	6,306,924	6,462,882	4,973,647
	6,409,901	6,541,348	5,104,839
number of presumably correct mappings	4,221,048	4,268,423	3,601,443
	3,931,308	3,954,435	3,280,134
	4,115,248	4,154,885	3,478,123
	4,219,686	4,265,874	3,592,822
	4,316,169	4,375,981	3,693,396

[†] with a single core of an AMD Opteron 8356 (2.3 GHz) and 128 GB of memory.

ten. In view of the above two possible reasons, (ii) seems to become dominant over (i) as the edit distance increases. This is because (i) is basically independent of the edit distance while (ii) occurs more frequently as the distance increases.

3.4 Performance Comparison with BFAST and BWA

We compared the performance of EEP with BFAST (in accurate mode with ten mask patterns) and BWA (with maximum edit distance 12 (`-n12`)) using actual biological data, HLA-targeted single-end SOLiD sequence data of length 50 bp. Large datasets, each with ten million reads, were used, which was advantageous for BFAST because it accessed a large amount of indexed data (121 GB). As shown in Table 2, EEP was as sensitive as BFAST and somewhat faster. BWA was much faster but much less sensitive than EEP and BFAST.

4 Conclusion

We have shown that efficient and sensitive genome mapping of short reads with higher error rates can be achieved by following the seed-and-extend paradigm and by using only continuous seeds, more precisely, for each read, by choosing a suitable set of variable-length continuous seeds allowing at most one or two errors according to the equi-entropy partitioning (EEP) method. They can be efficiently computed by using the Burrows-Wheeler transform. Comparison with BFAST, a highly sensitive genome mapping tool for short reads using spaced-seeds, demonstrated that our method is as sensitive as BFAST (in accurate mode with ten mask patterns) and somewhat faster using 1/4 less memory for mapping and 1/30 less disk space for indexed data in mapping 50-bp single-end SOLiD reads to the human genome.

References

1. Adjeroh, D., et al.: The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching. Springer Science+Business Media, LLC, Heidelberg (2008)
2. Breu, H.: A theoretical understanding of 2 base color codes and its application to annotation, error detection, and error correction. White Paper, Applied Biosystems (2010), publication 139WP01-02 CO13982
3. Burrows, M., Wheeler, D.: A block-sorting lossless data compression algorithm. Tech. Rep. 124, Digital Equipment Corporation (1994)
4. Clark, D.: Compact Pat Trees. Ph.D. thesis, the University of Waterloo (1996)
5. Ferragina, P., Manzini, G.: Opportunistic data structures with applications. In: Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science, pp. 390–398 (2000)
6. González, R., et al.: Practical implementation of rank and select queries. In: Proceedings of the 4th International Workshop on Experimental and Efficient Algorithms (WEA 2005), pp. 27–38. CTI Press and Ellinika Grammata, Greece (2005)
7. Holt, R., Jones, S.: The new paradigm of flow cell sequencing. Genome Research 18(6), 839–846 (2008)
8. Homer, N., Merriman, B., Nelson, S.: BFAST: An alignment tool for large scale genome resequencing. PLoS ONE 4(11), e7767 (2009)
9. Kimura, K., et al.: Computation of rank and select functions on hierarchical binary string and its application to genome mapping problems for short-read DNA sequences. Journal of Computational Biology 16(11), 1601–1613 (2009)
10. Langmead, B., et al.: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10(3), R25 (2009)
11. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25(14), 1754–1760 (2009)
12. Li, H., Ruan, J., Durbin, R.: Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Research 18(8), 1851–1858 (2008)
13. Li, M., et al.: PatternHunter II: Highly sensitive and fast homology search. In: Proceedings of the 14th International Conference on Genome Informatics 2003. Genome Informatics Series, pp. 164–175. World Scientific, Singapore (2003)
14. Li, R., et al.: SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics 25(15), 1966–1967 (2009)
15. Lippert, R.: Space-efficient whole genome comparisons with Burrows-Wheeler transforms. Journal of Computational Biology 12(4), 407–415 (2005)
16. Ma, B., Tromp, J., Li, M.: PatternHunter: faster and more sensitive homology search. Bioinformatics 18(3), 440–445 (2002)
17. Myers, G.: A fast bit-vector algorithm for approximate string matching based dynamic programming. Journal of the ACM 46(3), 395–415 (1999)
18. Schwartz, S., Kent, W., et al.: Humanmouse alignments with BLASTZ. Genome Research 13, 103–107 (2003)
19. Trapnell, C., Salzberg, S.: How to map billions of short reads onto genomes. Nature Biotechnology 27(5), 455–457 (2009)

A Practical Algorithm for Ancestral Rearrangement Reconstruction

Jakub Kováč[1], Broňa Brejová[1], and Tomáš Vinař[2]

[1] Department of Computer Science, Faculty of Mathematics, Physics, and Informatics, Comenius University, Mlynská Dolina, 842 48 Bratislava, Slovakia
kuko@ksp.sk, brejova@dcs.fmph.uniba.sk

[2] Department of Applied Informatics, Faculty of Mathematics, Physics, and Informatics, Comenius University, Mlynská Dolina, 842 48 Bratislava, Slovakia
vinar@fmph.uniba.sk

Abstract. Genome rearrangements are a valuable source of information about early evolution, as well as an important factor in speciation processes. Reconstruction of ancestral gene orders on a phylogeny is thus one of the crucial tools contributing to understanding of evolution of genome organization. For most models of evolution, this problem is NP-hard.

We have developed a universal method for reconstruction of ancestral gene orders by parsimony (PIVO) using an iterative local optimization procedure. Our method can be applied to different rearrangement models. Combined with a sufficently rich model, such as the double cut and join (DCJ), it can support a mixture of different chromosomal architectures in the same tree. We show that PIVO can outperform previously used steinerization framework and achieves better results on real data than previously published methods.

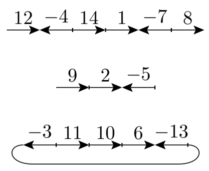
Datasets, reconstructed histories, and the software can be downloaded at http://compbio.fmph.uniba.sk/pivo/

1 Introduction

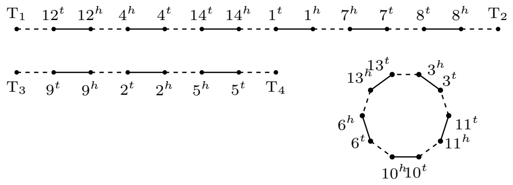
Genome rearrangements, such as inversions, transpositions, chromosome fusions and fissions, are evolutionary events that change the order of genes in genomes. These events are rare compared to point mutations, and are thought to be precursors of speciation [1]. Due to their low evolutionary rates, rearrangements are a valuable source of information about early evolution, and various rearrangement-based distances can provide phylogenetic signal well beyond the saturation of traditional nucleotide and protein point mutation models [2].

On the other hand, rearrangements make comparative analysis much more complex. Even though it is possible to reconstruct ancestral sequences in regions without breakpoints [3], it is much more difficult to order individual segments of ancestral genomes correctly [4].

In this paper, we present a new method for analysis of rearrangement histories, using gene orders of present-day species on a given phylogeny. Our goal is to reconstruct gene orders in all ancestral nodes of the phylogenetic tree.



(a) Genome π. Arrows represent markers with known orientation.



T1 12t 12h 4h 4t 14t 14h 1t 1h 7h 7t 8t 8h T2
T3 9t 9h 2t 2h 5h 5t T4
13h 13t 3h 3t 11t 11h 10t 10h 6t 6h

(b) Graph $G(\pi)$. Each marker consists of two extremities – head and tail; T denotes a telomere. Solid edges connect extremities of a single marker, dashed edges are adjacencies.

Fig. 1. Genome π consisting of two linear chromosomes and one circular chromosome (left) and its graph representation (right)

In particular, we are looking for the most parsimonious ancestral gene orders, minimizing the overall rearrangement distance (this problem is also called the *small phylogeny problem*). Even though our algorithm is not guaranteed to find the optimal solution, it presents a framework that generalizes most search strategies applied to various versions of this problem to date, e.g. [5,6].

Our method, PIVO (Phylogeny by IteratiVe Optimization), is one of the first practical tools applicable to analysis of real datasets spanning a complex phylogeny and accommodating a variety of genome architectures (single vs. multiple chromosomes, linear vs. circular chromosomes). In our experiments, we use the *double cut and join* (DCJ) model [7,8] for measuring rearrangement distance, but our method can be easily applied to other rearrangement distance measures reviewed in the next section. We demonstrate the accuracy of our program on the well-studied dataset of *Campanulaceae* chloroplast genomes [9], and apply it to the reconstruction of rearrangement histories of newly sequenced mitochondrial genomes of pathogenic yeasts from *Hemiascomycetes* clade [10].

Preliminaries, definitions, and related work. We will represent a genome as a set of *markers* (e.g., genes or synteny blocks) with known order and orientation. In this setting, a genome can be represented as a graph in which each (oriented) marker corresponds to two vertices, called *extremities* of the marker; the ends of linear chromosomes are represented by special vertices called *telomeres*. The edge set of this graph consists of the *marker edges*, joining the two extremities of each marker, and the *adjacencies*, joining two consecutive extremities in the genome or an extremity with a telomere (Fig. 1). Each connected component of this graph is thus a cycle or a path (representing a circular or a linear chromosome, respectively).

In our program, we employ the double cut and join (DCJ) [7,8] and reversal rearrangement model [11,12]. The DCJ model encompasses a rich set of rearrangement operations, and is able to represent both linear and circular chromosomes. An evolutionary operation in the DCJ model takes two adjacencies, $\{p, q\}$ and $\{r, s\}$, and replaces them by either $\{p, r\}$ and $\{q, s\}$, or $\{p, s\}$ and

$\{q, r\}$. This operation is quite general. A single DCJ operation can represent a reversal, translocation, chromosome fusion or fission, and excision or integration of a circular chromosome. Two operations can simulate a transposition. The DCJ distance is defined as the minimum number of DCJ operations needed to transform one genome into another. The distance between two genomes can be computed in linear time [8].

The *reversal model* is another popular model of evolution by rearrangements [13]. Each genome is represented by a *signed permutation* (a sequence of chromosomal markers with their orientations), allowing only genomes with a single linear chromosome. Genomes can be modified by a reversal operation that takes a contiguous section of markers in the permutation and replaces it with the markers in reversed order and with reversed orientations. The distance between two genomes with the same marker content is then the minimum number of reversals needed to transform one genome into the other. Reversal distance can be computed in linear time [14], and can be easily adapted for circular genomes. The *Hannenhalli-Pevzner (HP) model* [15] generalizes the reversal model to genomes with multiple linear chromosomes, allowing in addition to reversals also translocations, fusions, and fissions. The HP-distance can also be computed in linear time [16,17] and can be seen as a special case of the DCJ model restricted to operations that do not create circular chromosomes.

For completeness, we also consider a simple *breakpoint distance* [18]. A *breakpoint* between two genomes is a pair of markers that are consecutive in one genome but not in the other. The number of breakpoints between the two genomes is called the breakpoint distance. While this measure does not correspond to a particular set of evolutionary operations, it is clear that various rearrangement operations generally increase the breakpoint distance between the genomes, unless they reuse already created breakpoints.

While we can compute distances in all of these models efficiently, finding the most parsimonious solutions in more complex scenarios involving multiple genomes is more difficult. Perhaps the simplest scenario is the *median problem*, where we are given three extant genomes π_1, π_2, π_3 and a rearrangement distance measure d, and our task is to compute a single ancestral genome, a median π_M, that would minimize the sum of distances to the extant genomes $d(\pi_1, \pi_M) + d(\pi_2, \pi_M) + d(\pi_3, \pi_M)$.

Median problem has been shown to be NP-hard for almost every considered rearrangement model (unichromosomal reversal distance [19], unichromosomal breakpoint distance [20,21], multichromosomal linear breakpoint distance [22], unichromosomal [19] and multichromosomal [22] DCJ distance) and is conjectured to be NP-hard for other rearrangement models. One notable exception is the breakpoint distance on multiple circular or mixed chromosomes for which the median can be computed in polynomial time [22].

On the other hand, practical algorithms were developed for particular models, allowing to find good solutions to the median problem in many instances. Blanchette, Bourque, and Sankoff [23,18] reduce the breakpoint median problem to the travelling salesman problem (TSP) and then use a branch-and-bound algorithm to solve the resulting instance of TSP exactly. Siepel and Moret [24] and Caprara [25] propose exact practical solutions for the reversal median.

Heuristic median solvers try to move the given genomes closer and closer to each other until they meet at an approximate median [6,26,27,28]. A similar heuristic was also implemented for the DCJ model by Adam and Sankoff [29], while Xu and Sankoff have recently developed a DCJ median solver that is exact yet fast in practical instances [30].

The median problem is a special case of the *small phylogeny problem*, where we are given multiple extant genomes and their phylogenetic tree, and our goal is to compute genomes of their ancestors. According to the parsimony principle, the best reconstruction is the one involving the smallest number of rearrangement operations.

More formally, let G be the set of all possible genomes on a particular set of markers according to the chosen rearrangement model and let d be a distance measure on G. We are given a phylogenetic tree $T = (V, E)$ with the set of leaves L. For each leaf, we are also given a genome of the corresponding species, i.e., we are given a function $g : L \to G$. An *evolutionary history* is a function $h : V \to G$ extending g to the internal (ancestral) vertices. Our goal is to find an evolutionary history h which minimizes the overall evolutionary distance in the tree $d(h) = \sum_{\{u,v\} \in E} d(h(u), h(v))$.

The small phylogeny problem is trivially NP-hard for most rearrangement distance measures, since it is a generalization of the median problem. A notable exception is the breakpoint distance, for which the complexity of the small phylogeny problem is unknown.

Perhaps the most popular method for solving the small phylogeny problem is the *steinerization* method [31]. The main idea is to iteratively improve the evolutionary history until a local optimum is reached. In each iteration, we go through all internal vertices v. We take an ancestral genome π_v and its three neighbours π_a, π_b, π_c, compute the median π_M of π_a, π_b, π_c, and replace π_v with π_M if it yields a better overall score. If no vertex can be improved by taking the median of its neighbours, we have found a locally optimal evolutionary history. This approach is implemented in `BPAnalysis` software [23,18] for the breakpoint model and in `GRAPPA` software [32,33,34] for both breakpoint and reversal models. The same approach was implemented for the DCJ model by Adam and Sankoff [29]. `MGR` [6], another small phylogeny solver for reversal model, uses the simple heuristic based on using operations bringing genomes closer to other genomes in the tree. Our new algorithm, presented in the next section, encompasses and extends all these existing approaches.

Note that rearrangement models can also be used to reconstruct phylogenetic trees based on the order of markers in genomes. Perhaps the easiest method is to compute distances between all pairs of extant genomes and then use traditional distance-based algorithms for phylogeny reconstruction [2]. Another option is to solve the full *large phylogeny problem*, where we are looking for both the phylogenetic tree and the evolutionary history h minimizing the overall evolutionary distance. The small phylogeny can be used as a subroutine in the large phylogeny

Algorithm 1. Iterative local optimization

```
  Data: evolutionary history $h$
  Result: local optimum
1 $s' \leftarrow score(h)$, $s \leftarrow \infty$ ;
2 while $s' < s$ do
3     cand $\leftarrow$ generate lists of candidates (neighbourhood of $h$);
4     $h \leftarrow best$(cand);
5     $s \leftarrow s'$, $s' \leftarrow score(h)$;
6 end
7 return $h$
```

problem solvers. For a small number of species, we can enumerate all possible trees and for each tree compute the small phylogeny score, as in `BPAnalysis` and `GRAPPA` [23,34]. Alternatively, we can use the *sequential addition* heuristics, where we start from a trivial three-species star phylogeny and build the tree iteratively by adding new species, and reconstructing ancestral genomes in each step as in `MGR` or `amGRP` [6,35]. Our new method can also be used in this context.

2 Methods

In this section, we introduce a new general approach to the small phylogeny problem based on iterative local optimization. The basic idea is that in each step, we propose multiple candidates for ancestral genomes in each internal node of the tree and choose the most parsimonious combination of the candidates by dynamic programming. We will formulate the method in general terms for any rearrangement distance measure d that can be efficiently computed.

Consider a phylogenetic tree $T = (V, E)$ with the set of leaves L and genomes of extant species $g : L \to G$. We start with some history h_0. For a particular history h and each internal vertex v, we propose a set of candidates cand(h, v). We define a neighbourhood of history h as the set of all possible combinations of candidate genomes $N(h) = \{h' \mid \forall v \in V : h'(v) \in \text{cand}(h, v)\}$. We then find the best history in the neighbourhood $N(h)$ by a dynamic programming algorithm. If the new history is better than the previous one, we take it and repeat the iteration. Otherwise, we have found a local minimum and the algorithm terminates. We repeat the local optimization several times starting from different histories h_0. Algorithm 1 summarizes the local optimization method.

Example #1: For each internal vertex v, the set of candidates cand(h, v) can be the set of all the genomes within the distance 1 from $h(v)$. The neighbourhood of h is then the set of all histories we can obtain from h by performing at most one operation to each ancestral genome. Note that the size of $N(h)$ is exponential in the number of internal vertices, but as we will see later, we will never have to enumerate the entire neighbourhood.

Example #2: The steinerization approach mentioned in Section 1 is a special case of our method. Here, cand$(h, v) = \{h(v)\}$ for all vertices except for one

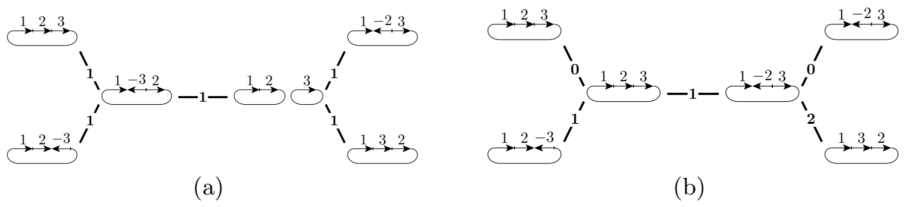


Fig. 2. A simple example showing a situation, where our more general approach outperforms the steinerization method. Given a quartet phylogeny and genomes at the leaves, the steinerization method has a local optimum with score 5 under the DCJ model (left), while there is a better solution with score 4 (right) which may be obtained when considering multiple candidates and choosing their best combination.

vertex w with neighbours a, b, c, for which $\text{cand}(h, w) = \{h(w), \text{median}(h(a), h(b), h(c))\}$.

Note that the opposite is not true, and in fact, proposing multiple candidates and then choosing the best combination is a crucial feature of our algorithm. Consider a simple example on a quartet phylogeny in Figure 2. The steinerization approach may get stuck in a local optimum as in Figure 2(a) (both ancestral genomes are medians of the neighbouring vertices). To avoid such local optima, the steinerization method is repeated from different starting configurations. On the other hand, if we consider *all* solutions of the median problems as candidates or if we consider the neighbouring genomes (that are within one DCJ operation from the current ancestors) and then choose the best combination, we obtain a better solution, shown in Figure 2(b).

We can generalize this example to configurations that will result in arbitrarily bad local optima of the steinerization method, whereas the global optimum can be found by our method.

2.1 Finding the Best History in a Neighbourhood

Even though the size of the neighbourhood $N(h)$ can be exponential (it has $\prod_v |\text{cand}(h, v)|$ elements), the best history can be found in polynomial time using dynamic programming.

Let c_i^u be the i-th candidate from $\text{cand}(h, u)$ and let $M[u, i]$ be the lowest score we can achieve for the subtree rooted at u if we choose candidate c_i^u as an ancestor. $M[u, i] = 0$ if u is a leaf. If u is an internal vertex with children v and w, we first compute values $M[v, j]$, $M[w, k]$ for all j, k. Then

$$M[u, i] = \min_j \{M[v, j] + d(c_i^u, c_j^v)\} + \min_k \{M[w, k] + d(c_i^u, c_k^w)\}.$$

This algorithm can be easily generalized for non-binary phylogenetic trees.

If n is the number of species, m the number of markers in each genome, and k the number of candidates for each ancestor, the best history can be found in time $O(nmk^2)$ (provided that the distance between two genomes can be computed in $O(m)$ time).

2.2 Strategies for Proposing Candidates

A crucial part of our method is proposing good candidates. By proposing more candidates, we explore a larger neighbourhood, but finding the best combination of candidates is slower. Furthermore, if we propose only candidates that are close to the genomes in the current history, the convergence to the local optimum may be slow. Here, we list several strategies for proposing candidates.

Extant species. In the initialization step, we can take genomes of the extant species as candidates in each internal node to get an evolutionary history to start with.

Intermediates. For a vertex v with adjacent vertices u and w, we can take intermediate genomes as candidates, i.e. if π, γ are genomes at u and w, we can sample genomes θ such that $d(\pi, \theta) + d(\theta, \gamma) = d(\pi, \gamma)$.

Medians. The steinerization method uses a median of the genomes in the three adjacent vertices as a candidate. Note that often there are many medians with the same score. Furthermore, Eriksen [36] shows that medians of moderately distant genomes may be spread wide apart. In our method, we do not need to decide beforehand which median to use. Instead, we consider all the medians as candidates, as already advocated by Eriksen [37] and Bernt *et al.* in `amGRP` [35] (`amGRP`, however, backtracks over different choices).

If we compute the median by branch-and-bound, the time to list all medians is comparable to the time to find just one median (median solvers of Siepel [24], and Caprara [25] are capable of listing all medians). If we try to find the median heuristically by repeatedly moving the given genomes closer to each other, we can take the intermediate genomes as candidates. Another option is to find just a single median and search its neighbourhood for medians which can be added to the candidate list.

Neighbours. We can include neighbourhoods of individual genomes. In particular, if $h(v) = \pi$, we can add the set $N(\pi) = \{\gamma \in G \mid d(\pi, \gamma) \leq 1\}$ to $\text{cand}(h, v)$. For most models, the size of $N(\pi)$ is roughly quadratic in the number of markers. Since for large genomes this becomes infeasible, we can include only genomes that do not increase the total distance to adjacent vertices, genomes closer to some genome in an adjacent vertex, or focus on a particular subset of neighbours.

Best histories. We can take several locally optimal histories and use the reconstructed ancestors as candidates. In this way we can "recombine" locally optimal solutions discovered previously.

2.3 Unequal Gene Content

A useful extension of our method is to allow a set of possible genomes in each leaf to be given on input instead of one fixed genome $g(v)$. This feature is useful if we are uncertain about the order of markers in some genomes. The algorithm will choose one of the alternative gene orders, so as to minimize the overall parsimony cost. Note that this choice can change between the iterations, and consequently

Table 1. The number of operations used to explain *Campanulaceae* dataset under different models and with different algorithms

	reversal distance	unichr. DCJ	general DCJ
`GRAPPA` (Moret *et al.* [33])	67		
`MGR` (Bourque and Pevzner [6])	65		
`GRAPPA` (Moret *et al.* [38])	64		
`BADGER` (Larget *et al.* [39])	64		
`ABC` (Adam and Sankoff [29])		64	**59**
`PIVO` (this paper)	**62**	**62**	**59**

we do not commit to a particular interpretation of the dataset until the end of the local optimization.

In addition to modeling uncertainty about the gene order in the extant species, we can also use this method for handling recent duplications or losses. Genome rearrangement models usually require equal gene content in all considered genomes. However, if one of the genomes contains a duplicated segment of markers, we can try to delete one or the other copy, producing two alternative gene orders that are used as candidates for the corresponding leaf of the tree. The algorithm will choose one of them for the locally optimal history h, presumably the one corresponding to the ancestral state before the duplication happened. We can extend this idea and use a larger set of candidates in case of multiple duplications or a gene loss. However, list of candidates will become prohibitively large for genomes with many such events.

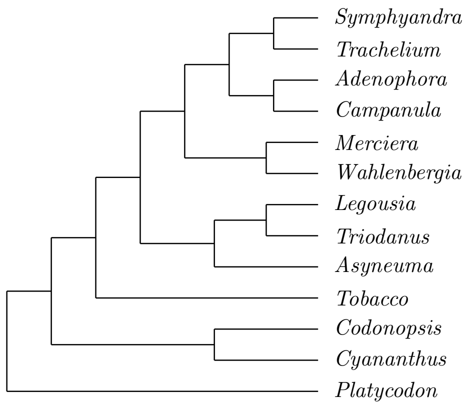
3 Results

We have implemented our new method and the strategies for exploring neighbourhoods described in the previous section using the DCJ and reversal rearrangement models. We demonstrate utility of our method on two datasets.

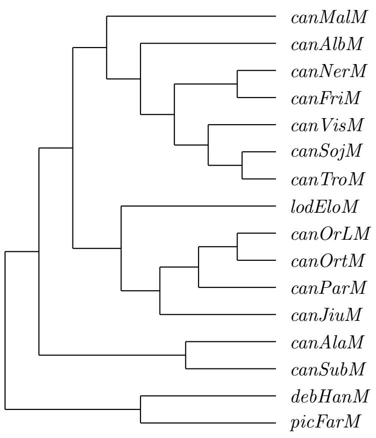
The *Campanulaceae* cpDNA dataset. For comparison, we applied our program to a well-studied dataset of 13 *Campanulaceae* chloroplast genomes [9]. Each genome in this dataset consists of a circular chromosome with 105 markers. Using the phylogenetic tree in Fig. 3(a) reconstructed by Bourque and Pevzner [6] with `MGR`, the results are presented in Table 1.

Using `GRAPPA` software, Moret *et al.* [33] found 216 tree topologies and evolutionary histories with 67 reversals. Bourque and Pevzner [6] using `MGR` later found a solution with 65 reversals. Even better solutions with 64 reversals were found by Moret *et al.* [38] (using `GRAPPA`) and Larget *et al.* [39] (using `BADGER`).

Adam and Sankoff [29] used the more general DCJ model and the phylogenetic tree by Bourque and Pevzner. They found a history with 64 DCJ operations with ancestors having a single chromosome, and a history with 59 DCJ operations with unconstrained ancestors. However, as Adam and Sankoff note: “There is no



(a) Phylogenetic tree of *Campanulaceae*



(b) Phylogenetic tree of *Candida*

Fig. 3. Phylogenetic trees used in the experiments

biological evidence in the *Campanulaceae*, or other higher plants, of chloroplast genomes consisting of two or more circles." The additional circular chromosomes are an artifact of the DCJ model, where a transposition or a block interchange operation can be simulated by circular excision and reincorporation.

We penalized multiple chromosomes in the dynamic programming objective function to avoid such artifacts and found several histories with 62 DCJ operations, where all the ancestors had a single circular chromosome. Moreover, these histories only require 62 reversals, which further improves on the best previously known result of 64 reversals by Moret *et al.* [38] and Larget *et al.* [39].

The *Hemiascomycetes* mtDNA dataset. We have also studied evolution of gene order in 16 mitochondrial genomes of pathogenic yeasts from the CTG clade of *Hemiascomycetes* [10]. The phylogenetic tree (Fig. 3(b)) was calculated by `MrBayes` [40] from protein sequences of 14 genes and is supported by high posterior probabilities on most branches.

The genomes consist of 25 markers (synteny blocks): 14 protein-coding genes, two rRNA genes, and 24 tRNAs. Several challenges make this dataset difficult. First, it combines genomes with a variety of genome architectures: *C. subhashii, C. parapsilosis*, and *C. orthopsilosis* are linear, *C. frijolesensis* has two linear chromosomes, and the rest of the species have circular-mapping chromosomes.

Some of the genomes (*C. albicans, C. maltosa, C. sojae, C. viswanathii*) contain recent duplications which cannot be handled by the DCJ model. As outlined in Section 2.3, we have removed duplicated genes, and included both possible forms of the genomes as alternatives in the corresponding leaves. Similarly, the genomes of *C. alai, C. albicans, C. maltosa, C. neerlandica, C. sojae*, and *L. elongisporus* contain long inverted repeats that are often subject to recombination resulting in reversal of the portion of the genome between the two repeats.

Both forms of the genome are routinely observed in the same species, and we include both of them in the corresponding leaf.

Finally, we penalized occurrences of multiple circular chromosomes and combinations of linear and circular chromosomes in ancestral genomes. Such combinations would likely represent artifacts of the DCJ model.

Our algorithm, using the *extant species, neighbours,* and *best histories* strategies, has found an evolutionary history with 78 DCJ operations. More detailed discussion of this dataset (including comparison to manual reconstruction in a subtree of closely related species) is included elsewhere [10].

4 Conclusion

We have developed a new method for reconstructing evolutionary history and ancestral gene orders, given the gene orders of the extant species and their phylogenetic tree. We have implemented our method using the double cut and join model and studied evolution of gene order in 16 mitochondrial yeast genomes, demonstrating applicability of our approach to real biological datasets. We have also analyzed the thoroughly studied *Campanulaceae* dataset and improved upon the previous results [33,6,38,39,29].

Our framework is compatible with a variety of rearrangement models and the optimization can be adjusted by introducing new strategies of generating candidate ancestral genomes. The use of the DCJ allowed us to study datasets that contained both linear and circular genomes and to contribute towards understanding mechanisms of genome linearization during the evolution [10].

In our experiments, we have explored only a small fraction of possible strategies offered by our framework. Systematic study of new initialization methods, candidate sets, and rearrangement measures may lead to even better results for a variety of practical problems. Our work also opened an avenue towards a systematic solution of the problems with unequal gene content. Similar directions can perhaps lead to the possibility of incorporating incompletely assembled genomes, a challenge posed by the next-generation sequencing technologies.

Acknowledgements. We would like to thank Jozef Nosek and Matúš Valach for helpful discussions on this problem and for providing *Hemiascomycetes* data. This research was supported by Comenius University Grant for Young Researchers UK/121/2011 to JK, Marie Curie grants IRG-224885 and IRG-231025 to TV and BB, and VEGA grant 1/0210/10.

References

1. Brown, K., Burk, L., Henagan, L., Noor, M.: A test of the chromosomal rearrangement model of speciation in Drosophila pseudoobscura. Evolution 58, 1856–1860 (2004)
2. Wang, L., Warnow, T., Moret, B., Jansen, R., Raubeson, L.: Distance-based genome rearrangement phylogeny. Journal of Molecular Evolution 63, 473–483 (2006)

3. Blanchette, M., Green, E., Miller, W., Haussler, D.: Reconstructing large regions of an ancestral mammalian genome in silico. Genome Research 14, 2412 (2004)
4. Ma, J., Zhang, L., Suh, B., Raney, B., Burhans, R., Kent, W., Blanchette, M., Haussler, D., Miller, W.: Reconstructing contiguous regions of an ancestral genome. Genome Research 16, 1557 (2006)
5. Moret, B., Warnow, T.: Toward new software for computational phylogenetics. Computer 35, 55–64 (2002)
6. Bourque, G., Pevzner, P.: Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Research 12, 26 (2002)
7. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics 21, 3340–3346 (2005)
8. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 163–173. Springer, Heidelberg (2006)
9. Cosner, M., Jansen, R., Moret, B., Raubeson, L., Wang, L., Warnow, T., Wyman, S.: An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In: Sankoff, D., Nadeau, J.H. (eds.) Comparative Genomics, pp. 99–121. Kluwer Academic Publishers, Dordrecht (2000)
10. Valach, M., Farkas, Z., Fricova, D., Kovac, J., Brejova, B., Vinar, T., Pfeiffer, I., Kucsera, J., Tomaska, L., Lang, B.F., Nosek, J.: Evolution of linear chromosomes and multipartite genomes in yeast mitochondria. Nucleic Acids Research (accepted, 2011)
11. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, pp. 178–189. ACM, New York (1995)
12. Bergeron, A., Mixtacki, J., Stoye, J.: The inversion distance problem. Math. of Evolution and Phylogeny, 262–290 (2005)
13. Kececioglu, J., Sankoff, D.: Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. Algorithmica 13, 180–210 (1995)
14. Bader, D.A., Moret, B.M.E., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. Journal of Computational Biology 8(5), 483–491 (2001)
15. Hannenhalli, S., Pevzner, P.A.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: Foundations of Computer Science (FOCS), pp. 581–592 (1995)
16. Bergeron, A., Mixtacki, J., Stoye, J.: HP distance via double cut and join distance. In: Ferragina, P., Landau, G.M. (eds.) CPM 2008. LNCS, vol. 5029, pp. 56–68. Springer, Heidelberg (2008)
17. Bergeron, A., Mixtacki, J., Stoye, J.: A new linear time algorithm to compute the genomic distance via the double cut and join distance. Theoretical Computer Science 410, 5300–5316 (2009)
18. Sankoff, D., Blanchette, M.: The median problem for breakpoints in comparative genomics. In: Jiang, T., Lee, D.T. (eds.) COCOON 1997. LNCS, vol. 1276, pp. 251–264. Springer, Heidelberg (1997)
19. Caprara, A.: The reversal median problem. INFORMS Journal on Computing 15, 93 (2003)
20. Pe'er, I., Shamir, R.: The median problems for breakpoints are NP-complete. Electronic Colloquium on Computational Complexity (ECCC) 5 (1998)

21. Bryant, D.: The complexity of the breakpoint median problem. Technical Report CRM-2579, CRM Universite de Montreal (1998)
22. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinformatics 10, 120 (2009)
23. Blanchette, M., Bourque, G., Sankoff, D.: Breakpoint phylogenies. In: Genome Informatics Workshop (GIW), pp. 25–34 (1997)
24. Siepel, A.C., Moret, B.M.E.: Finding an optimal inversion median: Experimental results. In: Gascuel, O., Moret, B.M.E. (eds.) WABI 2001. LNCS, vol. 2149, pp. 189–203. Springer, Heidelberg (2001)
25. Caprara, A.: On the practical solution of the reversal median problem. In: Gascuel, O., Moret, B.M.E. (eds.) WABI 2001. LNCS, vol. 2149, pp. 238–251. Springer, Heidelberg (2001)
26. Arndt, W., Tang, J.: Improving reversal median computation using commuting reversals and cycle information. Journal of Computational Biology 15(8), 1079–1092 (2008)
27. Yue, F., Zhang, M., Tang, J.: Phylogenetic reconstruction from transpositions. BMC Genomics **9** 9, S15 (2008)
28. Rajan, V., Xu, A., Lin, Y., Swenson, K., Moret, B.: Heuristics for the inversion median problem. BMC Bioinformatics 11, S30 (2010)
29. Adam, Z., Sankoff, D.: The ABC of MGR with DCJ. Evolutionary Bioinformatics Online 4, 69–74 (2008)
30. Xu, A.W., Sankoff, D.: Decompositions of multiple breakpoint graphs and rapid exact solutions to the median problem. In: Crandall, K.A., Lagergren, J. (eds.) WABI 2008. LNCS (LNBI), vol. 5251, pp. 25–37. Springer, Heidelberg (2008)
31. Sankoff, D., Cedergren, R., Lapalme, G.: Frequency of insertion-deletion, transversion, and transition in the evolution of 5S ribosomal RNA. Journal of Molecular Evolution 7, 133–149 (1976)
32. Moret, B., Wyman, S., Bader, D., Warnow, T., Yan, M.: A new implementation and detailed study of breakpoint analysis. In: Pacific Symposium on Biocomputing (PSB), pp. 583–594. Citeseer (2001)
33. Moret, B., Wang, L., Warnow, T., Wyman, S.: New approaches for reconstructing phylogenies from gene order data. Bioinformatics 17, S165 (2001)
34. Moret, B., Tang, J., Wang, L., Warnow, T.: Steps toward accurate reconstructions of phylogenies from gene-order data. Journal of Computer and System Sciences 65, 508–525 (2002)
35. Bernt, M., Merkle, D., Middendorf, M.: Using median sets for inferring phylogenetic trees. Bioinformatics 23, e129 (2007)
36. Eriksen, N.: Reversal and transposition medians. Theoretical Computer Science 374, 111–126 (2007)
37. Eriksen, N.: Median clouds and a fast transposition median solver. In: International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC), DMTCS Proceedings, pp. 373–384 (2009)
38. Moret, B., Siepel, A., Tang, J., Liu, T.: Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 521–536. Springer, Heidelberg (2002)
39. Larget, B., Kadane, J., Simon, D.: A Bayesian approach to the estimation of ancestral genome arrangements. Molecular Phylogenetics and Evolution 36, 214–223 (2005)
40. Ronquist, F., Huelsenbeck, J.: MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19, 1572 (2003)

Bootstrapping Phylogenies Inferred from Rearrangement Data

Yu Lin[⋆], Vaibhav Rajan[⋆], and Bernard M.E. Moret

Laboratory for Computational Biology and Bioinformatics, EPFL,
EPFL-IC-LCBB INJ230, Station 14, CH-1015 Lausanne, Switzerland
{yu.lin,vaibhav.rajan,bernard.moret}@epfl.ch

Abstract. Large-scale sequencing of genomes has enabled the inference of phylogenies based on the evolution of genomic architecture, under such events as rearrangements, duplications, and losses. Many evolutionary models and associated algorithms have been designed over the last few years and have found use in comparative genomics and phylogenetic inference. However, the assessment of phylogenies built from such data has not been properly addressed to date. The standard method used in sequence-based phylogenetic inference is the bootstrap, but it relies on a large number of homologous characters that can be resampled; yet in the case of rearrangements, the entire genome is a single character. Alternatives such as the jackknife suffer from the same problem, while likelihood tests cannot be applied in the absence of well established probabilistic models.

We present a new approach to the assessment of distance-based phylogenetic inference from whole-genome data; our approach combines features of the jackknife and the bootstrap and remains nonparametric. For each feature of our method, we give an equivalent feature in the sequence-based framework; we also present the results of extensive experimental testing, in both sequence-based and genome-based frameworks. Through the feature-by-feature comparison and the experimental results, we show that our bootstrapping approach is on par with the classic phylogenetic bootstrap used in sequence-based reconstruction, and we establish the clear superiority of the classic bootstrap and of our corresponding new approach over proposed variants. Finally, we test our approach on a small dataset of mammalian genomes, verifying that the support values match current thinking about the respective branches.

Our method is the first to provide a standard of assessment to match that of the classic phylogenetic bootstrap for aligned sequences. Its support values follow a similar scale and its receiver-operating characteristics are nearly identical, indicating that it provides similar levels of sensitivity and specificity. Thus our assessment method makes it possible to conduct phylogenetic analyses on whole genomes with the same degree of confidence as for analyses on aligned sequences. Extensions to search-based inference methods such as maximum parsimony and maximum likelihood are possible, but remain to be thoroughly tested.

1 Introduction

Large-scale sequencing of whole genomes has enabled the inference of phylogenies based on the evolution of genomic architecture, under such events as rearrangements,

[⋆] Who contributed equally to this work.

duplications, and losses. Many evolutionary models and associated algorithms have been designed over the last few years and have found use in comparative genomics and phylogenetic inference (see [12,24,36]). However, the assessment of phylogenies built from such data has not been properly addressed to date. The standard method used in sequence-based phylogenetic inference is the bootstrap [7,10]. It relies on the presence of a large number of homologous characters that can be resampled; yet in the case of rearrangements, the entire genome is a single character. Alternatives such as the jackknife suffer from the same problem, while likelihood ratio tests [2,13] cannot be applied in the absence of well established probabilistic models. Two preliminary approaches have been proposed, one based on the jackknife [32] and one based on random perturbations [19], but both fall short of the performance standard of the bootstrap on sequence data.

We describe a novel approach to the assessment of distance-based phylogenetic inference from whole-genome data. Our approach restates the main characteristics of the jacknife and bootstrap in terms of noise shaping, itself a longstanding approach to robustness assessment in engineering. For each feature of our method, we give an equivalent feature in the sequence-based framework and present the results of extensive experimental testing, in both sequence-based and genome-based frameworks, demonstrating that our bootstrapping approach for whole-genome data is on par with the classic phylogenetic bootstrap used in sequence-based reconstruction. We also establish the clear superiority of the classic bootstrap and of our corresponding new approach over proposed variants. (While the systematics community has long used the bootstrap as its reference method and has gained confidence in its use, no systematic experimental study of the approach had been conducted; our results fill this gap for distance-based methods and confirm the validity of phylogenetic bootstrapping.) Finally, we test our approach on a small dataset of mammalian genomes, verifying that the support values match current thinking about the respective branches.

The focus on distance-based methods is due in part to simplicity and convenience: by reducing the input genomes to a distance matrix, these methods not only simplify the characteristics of the input data, but also make it straightforward to compare our method with methods for sequence-based inference. The focus is also due in part to two other characteristics of distance-based methods: they are very efficient compared to optimization searches such as maximum parsimony and maximum likelihood; and they remain the most commonly used in routine phylogenetic reconstruction—neighbor-joining [29] and minimum evolution [5] account for nearly half of the citations to phylogenetic methods. But most of all the focus is justified by a unique characteristic of whole-genome data: under a model of rearrangements, duplications, and losses, it is possible to compute very precise maximum-likelihood estimates of the true evolutionary distance, as we have a shown in a series of papers [22,18,34,20]. Moreover, such distance estimates can be extended to take into account non-uniform distributions of the rearrangements (such as a preponderance of events affecting short segments of the genome), whereas we have almost no results about the combinatorial models of rearrangements when such rearrangements are not uniformly distributed. Finally, we have been able to extend this bootstrapping approach to inferences made under the Maximum Parsimony or Maximum Likelihood criteria, although the quality of the assessment provided by our approach for such settings remains to be established.

2 Background

We briefly review the bootstrap and jackknife approaches, as well as relevant characteristics of rearrangement data and of distance-based methods for phylogenetic inference.

2.1 Bootstrap and Jackknife

Given n data points $X = \{x_1, \ldots, x_n\}$ and a statistical estimator $E(x_1, \ldots, x_n)$, a *bootstrap replicate* is a fictional dataset $Y = \{y_1^*, \ldots, y_n^*\}$ constructed by sampling with replacement from X. From each such fictional dataset a value of the estimator E can be obtained. The key idea of bootstrapping is that the distribution of values thus obtained closely matches the original distribution of E and can be used to estimate the confidence limits on the estimator. The advantage of the method lies in its applicability to arbitrary and complicated estimators that may be analytically intractable [6,7].

In phylogeny reconstruction, the standard bootstrap for sequence data [10,11] samples columns with replacement from a multiple sequence alignment to create a new alignment matrix of identical dimensions. Thus each bootstrap replicate contains the same number of species and the same number of columns per species, but some columns from the original alignment may be duplicated and others omitted. Each column can be viewed as a variable that is drawn from a space of 4^s possible outcomes at each site—assuming nucleic acid sequence data with s species and neglecting insertions, deletions, and ambiguity codes. From each replicate, a tree can be reconstructed using any of the available reconstruction techniques (such as distance-based methods, maximum parsimony, or maximum likelihood). The tree thus obtained from a single bootstrap replicate is a *bootstrap tree*. Many bootstrap trees are generated through repeated sampling and the *bootstrap score* (or *support*) of a branch in the inferred tree is computed as the proportion of the bootstrap trees that contain this branch (viewed as a bipartition of leaves). Soltis and Soltis [33] and Holmes [15] discuss the pros and cons of the approach in phylogeny reconstruction.

A *jackknife* leaves out one observation at a time, thus creating a sample set $X_{(i)} = \{x_1, \ldots x_{i-1}, x_{i+1}, \ldots, x_n\}$. The estimator can be calculated on this new sample. The jackknife often provides a good approximation to the bootstrap, but it fails when the estimator is not smooth; moreover, the number of distinct sample sets is limited to the number of observations. Shao *et al.* [31] found that the generalized "delete-d" jackknife works well in practice, even for non-smooth estimators; in this version, d (or some fixed percentage) of the observations are randomly chosen and omitted to create the new sample set. A special case is *parsimony jackknifing* [8] in which an observation is omitted with fixed probability of $1/e$ when creating a new sample set. In such a case, the expected size of the new sample set is $(1 - 1/e)$ times the size of the original set, which corresponds to a modified bootstrapping procedure in which, after sampling, duplicate samples are not added to the new sample set.

No systematic comparison of these methods has been conducted in the context of phylogeny reconstruction. Felsenstein [10] hinted at the equivalence of support values from classic bootstrapping and from 50% jackknifing. Farris *et al.* [9] argued that 50% jackknifing deletes too many characters and does not allow one to maintain a useful relationship between group frequency and support; they advocated the use of parsimony jackknifing. Salamin *et al.* [30] compared bootstrapping and jackknifing in the context of maximum-parsimony reconstruction and reported that bootstrapping and 50%-jackknifing were comparable at confidence levels of 90% and higher. Finally, Mort *et al.* [25] compared bootstrapping with 50% and 33% jackknifing (with and without branch swapping) and reported that all three methods provide similar support values.

2.2 Rearrangement Data

Rearrangement data for a genome consists of lists of syntenic blocks (genes are an example) in the order in which they are placed along one or more chromosomes. Each syntenic block is identified by a marker, which is shared with all (or most) of its homologs in the genomes under study; for convenience, distinct markers are indexed arbitrarily from 1 to n. If every marker is shared and unique, the data is assumed to have been produced solely through rearrangements; otherwise, duplications and losses of syntenic blocks form another part of the evolutionary history. A chromosome (linear or circular) is represented by a signed permutation (linear or circular) of the markers' indices; the sign represents the strandedness of the corresponding syntenic block. A genome is a collection of such permutations, one per chromosome. Note that the actual sequence content of a syntenic block is ignored at this level: it was used only to identify the block. Interest in this type of data comes in part from the hypothesis that large-scale structural changes to the genome are "rare genomic changes" [28] and thus may clarify distant or problematic relationships among organisms. For that reason, such data has been used in a number of phylogenetic studies—see [24] for references.

2.3 Distance-Based Phylogeny Reconstruction from Rearrangement Data

Distance-based reconstruction methods typically run in time polynomial in the number and size of genomes—and fast and accurate heuristics exist for those where the scoring function cannot be computed in polynomial time, such as least-squares or minimum evolution methods. Further, methods like Neighbor-Joining (NJ) [29] provably return the true tree when given true evolutionary distances. However, the distances that can be computed with sequence data are often far from the true evolutionary distances, particularly on datasets with markedly divergent genomes. The true evolutionary distance—the actual number of evolutionary events between the two genomes—is impossible to measure, but it can be estimated using statistical techniques. A statistical model of evolution is used to infer an estimate of the true distance by deriving the effect of a given number of changes in the model on the computed measure and (algebraically or numerically) inverting the derivation to produce a maximum-likelihood estimate of the true distance under the model. This second step is often called a *distance correction* and has long been used for sequence data [35] as well as, more recently, for rearrangement data [23, 37, 39]. For multichromosomal genomes, we described such an estimator assuming equal "gene" content [18]. As rearrangement data is typically given in terms of syntenic blocks rather than genes, and as syntenic blocks are often unique, the limitation to equal gene content is not severe. Moreover, we recently refined our estimator to include gene duplication and loss events [20].

3 Robustness Estimation for Trees Reconstructed from Rearrangement Data

Providing bootstrap support scores is standard practice in phylogenetic reconstruction from sequence data. However, the classic bootstrap cannot be applied directly to rearrangement data because the collection of permutations forms a single character—a single rearrangement or duplication can affect an entire permutation or even several of them. For example, sampling genes with replacement from leaf genomes is infeasible. In the world of sequence data this is equivalent to an alignment with a single column, albeit one where each character can take any of a huge number of states. Two approaches have been suggested in the past, but do not match the accuracy of the bootstrap. One, proposed by Shi *et al.* [32], is a jackknifing technique; the second, proposed by us, uses perturbations in the permutations with known effects on the distance [19].

We design different methods for rearrangement data and devise analogous methods for sequence data (if they do not exist) and vice versa. We study their behavior with both kinds of data with the aim of developing a method for rearrangement data that is as successful as the classic bootstrap is for sequence data. For a method M that operates on sequence data, we denote by M^* the corresponding method for rearrangement data; we use regular font to denote existing methods, bold font to denote the new methods described in this paper.

The methods we present here for rearrangement data rely on our distance estimator [18] and so must be used with distance-based reconstruction methods. Our distance estimator computes the estimated true distance between two multichromosomal genomes, based only on the number of shared adjacencies and the number of linear chromosomes in each genome. This limited view of the input data is crucial, as many of the sampling approaches we describe below do not produce valid genome permutations (e.g., because of additional copies of adjacencies), yet still allow us to tally the number of linear chromosomes and of shared adjacencies.

Our robustness estimator based on distance perturbation [19], hereafter denoted BP*, permutes each leaf genome through a (randomly chosen) number of random rearrangements, estimates the new pairwise distances, then subtracts from each pairwise estimate the number of rearrangement operations applied to each of the two genomes. Thus it relies on additivity, a property likely to be respected with rearrangement data due to its huge state space. We can design an equivalent for sequence data: for each sequence, apply some random number of randomly chosen mutations, then estimate every pairwise distances, and finally subtract from that estimate the number of mutations applied in the perturbation step to each of the two sequences—a method we denote **BP**. **BP** is less reliable than BP*, as it is much more likely that some of the mutations used in the perturbations cancel each other or cancel some of the mutations on the edit path between the two sequences.

We can view the classic bootstrap for sequence data (hereafter denoted BC) in terms of noise generation. The original multiple sequence alignment gives rise to a distance matrix D. Each replicate dataset created by sampling columns with replacement from the alignment also gives its corresponding matrix B of perturbed pairwise distances. An entry of the replicate matrix corresponding to leaves i and j can thus be written as $B(i,j) = D(i,j) + N(i,j)$ where $N(i,j)$ denotes the perturbation in the distance introduced by the resampling. This noise parameter is hard to characterize exactly, but it leads us to define bootstrapping approaches based on producing increasingly refined estimates of the noise. (In that sense, BP* and **BP** attempt to shape the noise by returning to the underlying evolutionary process of rearrangement or mutation.)

Bootstrapping by adding Gaussian Noise (hereafter denoted **BGN**), adds Gaussian noise of mean 0 to each entry in the distance matrix. The standard deviation is empirically determined to match as well as possible the noise added by BC. Since the noise added during the sampling process in BC is not random, this is a very rough estimate, but a useful comparison point. In the replicate matrices produced by BC, the noise $N(i,j)$ depends on the pairwise distance $D(i,j)$, so the next step is to design a bootstrap method based on pairwise comparisons, hereafter denoted **BPC**. The bootstrap matrix $B(i,j)$ for **BPC** is constructed by calculating the perturbed pairwise distance for each pair: for each pair of sequences i,j, we construct a new pair of sequences i',j' by sampling columns with replacement, where each column has only two characters and set $B(i,j) = D(i',j')$.

An equivalent method **BPC*** can be designed for rearrangement data, albeit with some complications. Since our distance estimator relies on the number of shared adjacencies, a natural choice is to sample adjacencies in the genome. While the evolution of a specific adjacency depends directly on several others, independence can be assumed if we assume that once an adjacency is broken during evolution it is not formed again—an analog of Dollo parsimony, but one that is very likely in rearrangement data due to the enormous state space. For each pair of genomes i,j, we construct two new pairs of genomes. We sample adjacencies from genome i with replacement and use only these adjacencies to compute the distance $D_1(i,j)$ of leaf i to leaf j. (Note that some adjacencies may be overcounted and some omitted.) Then we sample adjacencies from genome j with replacement and use only these adjacencies to compute the distance $D_2(i,j)$ of leaf j to leaf i. Finally, we set $B(i,j) = (D_1(i,j) + D_2(i,j))/2$.

The noise $N(i,j)$ may depend not just on the pairwise distance $D(i,j)$, but also on other distances in the tree, since BC samples columns with replacement for all leaf sequences *at once*. The next step in modeling $N(i,j)$ is thus to sample from all adjacencies (including telomeres). The total number of possible adjacencies (including telomeres) for n syntenic blocks is roughly $2n^2$, but in a given genome there are at most $2n$ adjacencies and each adjacency conflicts with at most $4n$ other adjacencies. Thus, for large genomes, we may assume that adjacencies are independent, just as columns of an alignment are assumed to be independent in BC. We can now mimic closely the sampling procedure of BC in a rearrangement context, producing procedure **BC***. From the list of all possible adjacencies, **BC*** samples with replacement to form a collection of adjacencies; only adjacencies in this collection are then considered in counting the number of shared adjacencies and then estimating the true evolutionary distances between genomes. (Note that some shared adjacencies are counted more than once due to the sampling with replacement.)

We know that classic bootstrapping (BC) is comparable in performance to parsimony jackknifing (which we denote PJ) in the sequence world. PJ is (asymptotically) equivalent to sampling with replacement (as in BC), but without overcounting, that is, when sampling gives a column that has been previously selected, it is not added to the replicate. Thus we can obtain the equivalent of PJ for rearrangement data, call it **PJ***: selected adjacencies are not counted more than once for computing the number of shared adjacencies between leaf genomes. Other versions of jackknifing are similarly easy to design. For instance, a $d\%-$jackknife (dJK) omits $d\%$ of the columns to create a replicate, so, from the set of all adjacencies (in all the leaf genomes) a $d\%$-jackknife (*d***JK***) deletes $d\%$ of the adjacencies at random and only the remaining adjacencies are used in estimating the true pairwise distances. In contrast, the previous jackknifing approach for rearrangement data, developed by Shi *et al.* [32], produces replicates by deleting syntenic blocks from the genome: a $d\%$-jackknife, in their method, produces a dataset where $d\%$ of the markers are randomly deleted from all leaf genomes. The authors recommend setting $d = 40$; we call the resulting method JG*. Note that our approach to jackknifing deletes adjacencies instead of markers.

In summary, we have designed a bootstrapping procedure, **BC***, that closely mimics the classic bootstrap for phylogenetic reconstruction, BC, and jackknifing procedures, *d***JK*** (including, as a special case, **PJ***), that closely mimic the $d\%-$jackknife (and parsimony jackknife PJ). Along the way, we have also designed less refined versions of bootstrapping and their equivalents for sequence data. In our experiments, we use all of these, plus JG*, the marker-based jackknifing approach of Shi *et al.*, plus BP*, our earlier approach based on introducing perturbations, in the permutations, of known effect on the distances [19]. A summary of all the methods can be found in table 1.

Table 1. A summary of all the methods

BGN, **BGN***	Bootstrap by adding Gaussian Noise to the distance matrix.
BPC, **BPC***	Bootstrap by Pairwise Comparisons: for each pair of sequences/genomes, sample columns/adjacencies with replacement to compute pairwise distance.
BC, **BC***	Classic Bootstrap: sample columns with replacement to obtain replicate; sample adjacencies with replacement to compute distance matrix.
PJ, **PJ***	Parsimony Jackknifing: choose each column with $1-1/e$ probablity to create replicate; sample adjacencies with replacement and discard duplicates to compute distance matrix.
dJK, **dJK***	$d\%$-JackKnife: Omit $d\%$ of columns at random to produce replicate; omit $d\%$ of adjacencies at random to compute distance matrix.
BP, BP*	Bootstrap by Perturbations: apply random mutations/rearrangements to get replicates.
JG*	Jackknife Genes: Marker based jackknifing method of Shi *et al* for rearrangement data.

4 Experimental Design

Our simulation studies follow the standard procedure in phylogeny reconstruction (see, e.g., [14]): we generate model trees under various parameter settings, then use each model tree to produce a number of true trees on which we evolve artificial genomes from the root down to the leaves to obtain datasets of leaf genomes for which we know the complete history. The sequences are evolved by random point mutations under the

Kimura 2-parameter (K2P) model (see [35]) using various transition/transversion ratios; the permutations are evolved through double-cut-and-join (DCJ) operations chosen uniformly at random. (DCJ [3, 41], has become the most commonly used model of rearrangement for multichromosomal data and is the rearrangement operator targeted by our distance estimator.) The resultant leaf sequences are without gaps, are of the same length and do not need further alignment. For distance-based reconstruction, the distances between leaf sequences are given by the standard distance estimate for the K2P model [35] and the tree is reconstructed with the Neighbor-Joining (NJ) algorithm [29]. For rearrangement data we reconstruct trees by computing a distance matrix using our DCJ-based true distance estimator [18] and then using this matrix as input to NJ. (We chose to use NJ rather than the better FastME [5] in order to highlight the discriminating ability of the bootstrapping methods.)

A model tree consists of a rooted tree topology and corresponding branch lengths. The trees are generated by a three-step process. We first generate birth-death trees using the tree generator in the software R [27], with a death rate of 0 and various birth rates (data shown below is for a rate of 0.001). The branch lengths in this tree are ultrametric (the root-to-leaf paths all have the same length), so, in the second step, the branch lengths are modified to eliminate the ultrametricity. Choosing a parameter c, for each branch we sample a number s uniformly from the interval $[-c, +c]$ and multiply the original branch length by e^s (we used various values of c; data shown below is for $c = 2$). Finally, we rescale branch lengths to achieve a target diameter D for the model tree. (Note that the unit of "length" is one expected evolutionary operation—rearrangement or mutation.) Each branch length now represents the *expected* number of evolutionary operations on that branch. From a single model tree, a set of trees is generated for simulation studies by retaining the same topology and varying the branch lengths by sampling, for each branch in the tree, from a Poisson distribution with a mean equal to that of the corresponding branch length in the model tree.

Experiments are conducted by varying the number of syntenic blocks and the target diameter. We use trees with 100 leaves. Among the many parameter values tested we show the following representative settings: for sequence data, each leaf has 10,000 characters and the tree diameter is 20,000, while, for rearrangement data, each genome has 5,000 markers and the tree diameter is 15,000. For each setting of the parameters, 100 model trees are generated and from each model tree 10 datasets are created; we then average results over the resulting 1,000 trees. For each experiment we produce 100 replicates and thus 100 bootstrap trees from which to compute the bootstrap support of each branch.

A Receiver-Operator-Characteristic (ROC) curve is drawn for every method we investigate. In this plot, a point is a particular bootstrapping test, defined by its *sensitivity* and *specificity*; in the system of coordinates of our figures, a perfect test would yield a point at the upper left-hand corner of the diagram, with 100% sensitivity and 100% specificity. Define E to be the set of edges in the true tree and T_t, for a threshold t, to consist of those edges in the inferred tree that are contained in more than t% of the bootstrap trees. Sensitivity is the proportion of true edges that are also in T_t, $|T_t \cap E|/|E|$, while specificity is the proportion of edges in T_t that are true edges, $|T_t \cap E|/|T_t|$. In our tests we use every fifth value in the range $[0, 100]$ as thresholds.

5 Experimental Results and Analysis

Fig. 1 shows the ROC curves of the methods for sequence data, for 100 sequences of 10,000 characters each, and a tree diameter of 20,000. The four “reference” methods—50%-jackknifing (50JK), classic bootstrapping (BC), $(1/e)$%-jackknifing (37JK), and parsimony jackknifing (PJ)—are nearly indistinguishable and clearly dominate the others. The analogs of all the other methods developed for rearrangement data (**BP**, **BPC** and **BGN**) are clearly worse than the above four, with **BP** and **BPC** being comparable and the most primitive noise-shaping method, **BGN**, doing the worst.

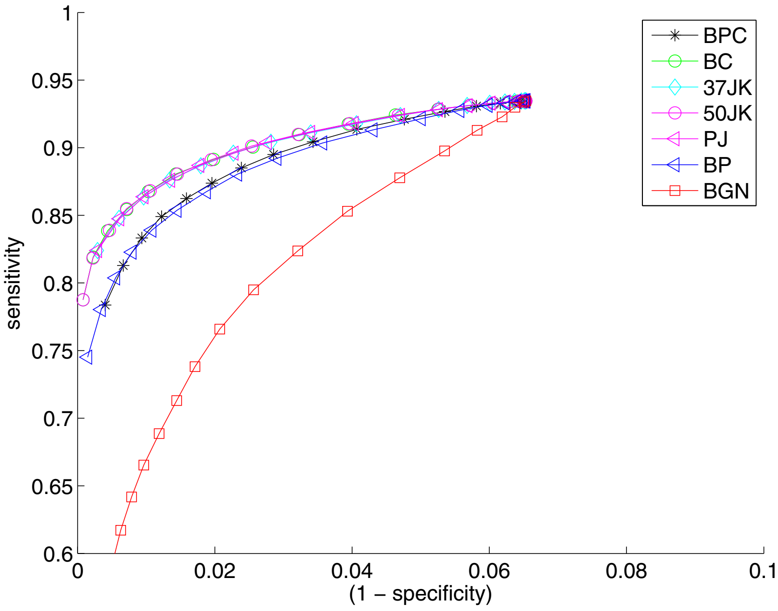


Fig. 1. Bootstrapping methods for sequence data

Fig. 2 shows the ROC curves for rearrangement data, for 100 genomes of 5,000 markers each, and a tree diameter of 15,000. The results follow the same pattern as for sequence data: **BC***, **PJ***, **50JK***, and **37JK*** are nearly indistinguishable and clearly dominate all others. They are followed by BP* and **BPC***, which are comparable, while the Gaussian noise approach, **BGN***, again does the worst. JG*, the marker-based jackknifing technique of Shi *et al.*, is better than **BGN***, but trails all other methods. The differences are particularly marked at very high levels of specificity; at 98% specificity, for instance, the top four methods retain nearly 90% sensitivity, but JG* drops to 80%. Very high specificity is the essential characteristic of a good bootstrap method.

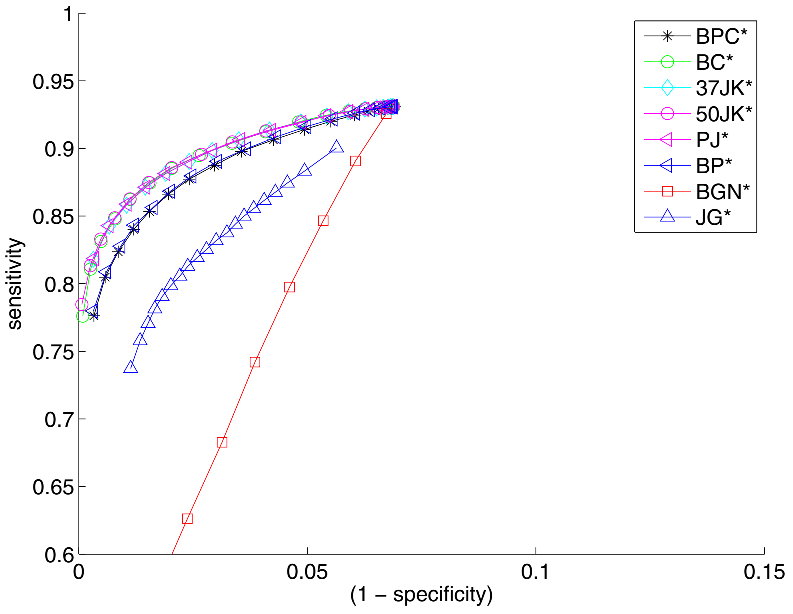


Fig. 2. Bootstrapping methods for rearrangement data

5.1 A Dataset of Vertebrate Genomes

We also tested our bootstrapping methods on a real dataset: the genomes of 10 species from the Ensembl Mercator/Pecan alignments with 8,380 common markers. Four of these genomes (horse, chimpanzee, rhesus, and orangutan) are not well assembled: their draft genomes have nearly twice as many contigs as there are chromosomes—but the effect on our adjacency-based distance estimator is minimal, given the large number of markers. Fig. 3 shows the inferred phylogeny and highlights the two edges with lowest bootstrap support (according to our **BC*** method). Based on previous studies [21, 26, 1, 17, 40, 4] the edge e_1 is uncertain: some studies place the primates in a clade with rodents, while others place them in a clade with the carnivores. Thus we would expect e_1 to receive the lowest support in the tree. **BC*** does give it the lowest support: 77% for e_1 and 83% for e_2. BP* gives low support values for both (49% for e_1 and 44% for e_2), but fails to identify e_1 as the least supported edge, while JG* erroneously gives high support values to both (100% for e_1 and 90% for e_2).

6 Discussion

Our new approach for whole-genome data, based on the sampling of adjacencies, matches the classic bootstrap and parsimony jackknife approaches and thus provides the first reliable method for assessing the quality of phylogenetic reconstruction from such data.

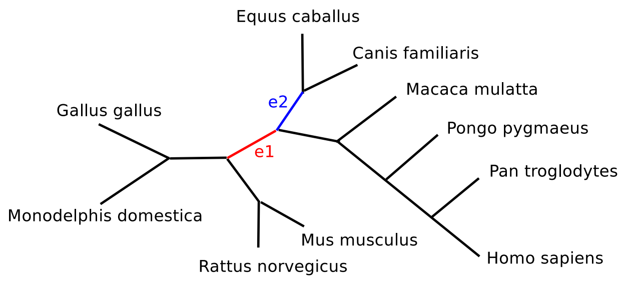


Fig. 3. Inferred phylogeny of 10 vertebrates

In the process of testing various methods, we also confirmed past findings about the superiority of the phylogenetic bootstrap and of the parsimony jackknife. Our results clearly indicate that duplicate samples play no role in the process—parsimony jackknifing works at least as well and occasionally slightly better. Indeed, the best sampling strategy appears to be a random sampling of half of the characters. Given the very high computational cost of the bootstrap, using half the number of characters in sequence-based analyses appears a worthwhile computational shortcut, especially as it delivers even better results.

Our study focuses on distance-based methods, which reduce the collection of input genomes to a distance matrix. Our basic approach is to equate sampling characters in sequence data with sampling adjacencies in whole-genome data. Any reconstruction method that can handle such data can use this bootstrap procedure. Our reconstruction method is one such method since our distance estimator only counts the number of shared adjacencies between genomes and the number of linear chromosomes in each of them. Possible alternatives for methods (such as Maximum Parsimony) that are unable to handle such data include parsimony jackknifing and direct encoding of adjacencies into sequences. In parsimony jackknifing (**PJ***), each original genome is represented by a set of contiguous regions in the bootstrap; if the reconstruction method can handle such inputs, then this is the best method. Encoding rearrangement data into sequences was proposed many years ago (see [38]) in two different versions (binary encodings and multistate encodings). In such methods, the input is simply a collection of (perfectly) aligned sequences and so the output can be assessed by the standard phylogenetic bootstrap. The early encodings fared poorly in comparison with MP methods (for rearrangement data), but a recent paper [16] suggests that a more complex encoding may overcome these problems.

References

1. Amrine-Madsen, H., Koepfli, K.P., Wayne, R., Springer, M.: A new phylogenetic marker, apolipoprotein b, provides compelling evidence for eutherian relationships. Mol. Phyl. Evol. 28(2), 225–240 (2003)
2. Anisimova, M., Gascuel, O.: Approximate likelihood-ratio test for branches: A fast, accurate, and powerful alternative. Syst. Biol. 55(4), 539–552 (2006)

3. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 163–173. Springer, Heidelberg (2006)
4. Cannarozzi, G., Schneider, A., Gonnet, G.: A phylogenomic study of human, dog, and mouse. PLoS Comput. Biol. 3, e2 (2007)
5. Desper, R., Gascuel, O.: Theoretical foundation of the balanced minimum evolution method of phylogenetic inference and its relationship to weighted least-squares tree fitting. Mol. Biol. Evol. 21(3), 587–598 (2003)
6. Efron, B., Gong, G.: A leisurely look at the bootstrap, the jackknife, and cross-validation. American Statistician 37(1), 36–48 (1983)
7. Efron, B., Tibshirani, R., Tibshirani, R.: An Introduction to the Bootstrap. Chapman & Hall/CRC, Boca Raton (1993)
8. Farris, J.: The future of phylogeny reconstruction. Zoologica Scripta 26(4), 303–311 (1997)
9. Farris, J., Albert, V., Källersjö, M., Lipscomb, D., Kluge, A.: Parsimony jackknifing outperforms neighbor-joining. Cladistics 12(2), 99–124 (1996)
10. Felsenstein, J.: Confidence limits on phylogenies: an approach using the bootstrap. Evol. 39, 783–791 (1985)
11. Felsenstein, J., Kishino, H.: Is there something wrong with the bootstrap on phylogenies? A reply to Hillis and Bull. Syst. Biol. 42(2), 193–200 (1993)
12. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. MIT Press, Cambridge (2009)
13. Guindon, S., Gascuel, O.: PHYML—a simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. 52(5), 696–704 (2003)
14. Hillis, D., Huelsenbeck, J.: Assessing molecular phylogenies. Science 267, 255–256 (1995)
15. Holmes, S.: Bootstrapping phylogenetic trees: theory and methods. Statistical Science 18(2), 241–255 (2003)
16. Hu, F., Gao, N., Tang, J.: Maximum likelihood phylogenetic reconstruction using gene order encodings. In: Proc. 8th IEEE Symp. Comput. Intell. in Bioinf. & Comput. Biol. (CIBCB 2011). IEEE Press, Los Alamitos (accepted, to appear 2011)
17. Huttley, G., Wakefield, M., Easteal, S.: Rates of genome evolution and branching order from whole-genome analysis. Mol. Biol. Evol. 24(8), 1722–1730 (2007)
18. Lin, Y., Moret, B.: Estimating true evolutionary distances under the DCJ model. In: Proc. 16th Int'l Conf. on Intelligent Systems for Mol. Biol. (ISMB 2008) (2008); Bioinformatics 24(13), i114–i122 (2008)
19. Lin, Y., Rajan, V., Moret, B.: Fast and accurate phylogenetic reconstruction from high-resolution whole-genome data and a novel robustness estimator. In: Tannier, E. (ed.) RECOMB-CG 2010. LNCS, vol. 6398, pp. 137–148. Springer, Heidelberg (2010)
20. Lin, Y., Rajan, V., Swenson, K., Moret, B.: Estimating true evolutionary distances under rearrangements, duplications, and losses. In: Proc. 8th Asia Pacific Bioinf. Conf. (APBC 2010) (2010); BMC Bioinformatics 11(suppl. 1), s54 (2010)
21. Madsen, O., et al.: Parallel adaptive radiations in two major clades of placental mammals. Nature 409, 610–614 (2001)
22. Marron, M., Swenson, K., Moret, B.: Genomic distances under deletions and insertions. Theor. Comput. Sci. 325(3), 347–360 (2004)
23. Moret, B., Tang, J., Wang, L.S., Warnow, T.: Steps toward accurate reconstructions of phylogenies from gene-order data. J. Comput. Syst. Sci. 65(3), 508–525 (2002)
24. Moret, B., Warnow, T.: Advances in phylogeny reconstruction from gene order and content data. In: Zimmer, E., Roalson, E. (eds.) Molecular Evolution: Producing the Biochemical Data, Part B. Methods in Enzymology, vol. 395, pp. 673–700. Elsevier, Amsterdam (2005)
25. Mort, M., Soltis, P., Soltis, D., Mabry, M.: Comparison of three methods for estimating internal support on phylogenetic trees. Syst. Biol. 49(1), 160–171 (2000)

26. Murphy, W., Eizirik, E., Johnson, W., Zhang, Y., Ryder, O., O'Brien, S.: Molecular phylogenetics and the origins of placental mammals. Nature 409, 614–618 (2001)
27. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (2009)
28. Rokas, A., Holland, P.: Rare genomic changes as a tool for phylogenetics. Trends in Ecol. and Evol. 15, 454–459 (2000)
29. Saitou, N., Nei, M.: The neighbor-joining method: A new method for reconstructing phylogenetic trees. Mol. Biol. Evol. 4, 406–425 (1987)
30. Salamin, N., Chase, M., Hodkinson, T., Savolainen, V.: Assessing internal support with large phylogenetic DNA matrices. Mol. Phyl. Evol. 27(3), 528 (2003)
31. Shao, J., Wu, C.: A general theory for jackknife variance estimation. Annals of Statistics 17(3), 1176–1197 (1989)
32. Shi, J., Zhang, Y., Luo, H., Tang, J.: Using jackknife to assess the quality of gene order phylogenies. BMC Bioinformatics 11(1), 168 (2010)
33. Soltis, P., Soltis, D.: Applying the bootstrap in phylogeny reconstruction. Statist. Sci. 18(2), 256–267 (2003)
34. Swenson, K., Marron, M., Earnest-DeYoung, J., Moret, B.: Approximating the true evolutionary distance between two genomes. In: Proc. 7th SIAM Workshop on Algorithm Engineering & Experiments (ALENEX 2005). SIAM Press, Philadelphia (2005)
35. Swofford, D., Olson, G., Waddell, P., Hillis, D.: Phylogenetic inference. In: Hillis, D., Moritz, C., Mable, B. (eds.) Molecular Systematics, 2nd edn., ch. 11. Sinauer Assoc. (1996)
36. Tannier, E.: Yeast ancestral genome reconstructions: The possibilities of computational methods. In: Ciccarelli, F.D., Miklós, I. (eds.) RECOMB-CG 2009. LNCS, vol. 5817, pp. 1–12. Springer, Heidelberg (2009)
37. Wang, L.S.: Exact-IEBP: a new technique for estimating evolutionary distances between whole genomes. In: Proc. 33rd Ann. ACM Symp. Theory of Comput. (STOC 2001), pp. 637–646. ACM Press, New York (2001)
38. Wang, L.S., Jansen, R., Moret, B., Raubeson, L., Warnow, T.: Fast phylogenetic methods for genome rearrangement evolution: An empirical study. In: Proc. 7th Pacific Symp. on Biocomputing (PSB 2002), pp. 524–535. World Scientific Pub., Singapore (2002)
39. Wang, L.S., Warnow, T.: Estimating true evolutionary distances between genomes. In: Gascuel, O., Moret, B.M.E. (eds.) WABI 2001. LNCS, vol. 2149, pp. 176–190. Springer, Heidelberg (2001)
40. Wildman, D., et al.: Genomics, biogeography, and the diversification of placental mammals. Proc. Nat'l. Acad. Sci., USA 104(36), 14395–14400 (2007)
41. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics 21(16), 3340–3346 (2005)

Speeding Up Bayesian HMM by the Four Russians Method

Md Pavel Mahmud[1] and Alexander Schliep[1,2]

[1] Department of Computer Science, Rutgers University, New Jersey, USA
[2] BioMaPS Institute for Quantitative Biology, Rutgers University, New Jersey, USA
{pavelm,schliep}@cs.rutgers.edu

Abstract. Bayesian computations with Hidden Markov Models (HMMs) are often avoided in practice. Instead, due to reduced running time, point estimates – maximum likelihood (ML) or maximum a posterior (MAP) – are obtained and observation sequences are segmented based on the Viterbi path, even though the lack of accuracy and dependency on starting points of the local optimization are well known. We propose a method to speed-up Bayesian computations which addresses this problem for regular and time-dependent HMMs with discrete observations. In particular, we show that by exploiting sequence repetitions, using the four Russians method, and the conditional dependency structure, it is possible to achieve a $\Theta(\log T)$ speed-up, where T is the length of the observation sequence. Our experimental results on identification of segments of homogeneous nucleic acid composition, known as the DNA segmentation problem, show that the speed-up is also observed in practice.

Availability: An implementation of our method will be available as part of the open source GHMM library from http://ghmm.org.

Keywords: Hidden Markov Model, Bayesian, MCMC, Gibbs Sampling, Compression, Four Russians, Speed-up, DNA Segmentation.

1 Introduction

Hidden Markov Models have been used extensively for sequence classification tasks in many areas including speech recognition [7], natural language processing [19], and bioinformatics [12]. For analyzing biological sequences, HMMs are particularly useful for example in sequence alignment problems [12], gene finding [8], CpG island detection [11], DNA segmentation [10,11,18,24], and promoter detection [25]. These application problems all lead to the computational task of segmenting the input, an observation sequence, based on the most likely assignment of hidden states.

For simplicity and efficiency reasons, point estimates such as maximum likelihood (ML) or maximum a posterior (MAP), computed with Baum-Welch [2] and variants, have traditionally been used for learning HMM parameters. Based on these estimates segmentations have been computed with the Viterbi path. This ignores uncertainty in model parameters and consequently predictions based on

ML or MAP trained models often turn out to be inferior in practice. In contrast, a full Bayesian approach integrates out model parameters and thus removes dependency on one parameter estimate to improve HMM based prediction. As closed form solutions are not available for HMMs, one frequently uses Markov Chain Monte Carlo (MCMC) sampling techniques like Gibbs sampling or Metropolis-Hastings [3] instead of integration. One particular form of Gibbs sampling for HMM, known as forward-backward Gibbs sampling [9,29], is popular in several communities [1,15,26,28,30,31] for its improved convergence rate through use of forward and backward recursions.

However, depending on the problem, forward-backward Gibbs sampling can still take many iterations to converge. Careful choice of prior distributions and corresponding hyper-parameters can sometimes increase the convergence rate but it remains computationally inefficient compared to using point estimates. One possible source of improvement is to make use of the characteristics of the observed sequence to speed-up computations.

Using text compression techniques (LZ78, byte pair encoding, four Russians, etc.), Mozes *et. al.* [17,21] have exploited repetitions in long biological sequence to improve the running time of the Viterbi algorithm which computes the most likely hidden state sequence given the observation sequence as well as forward, backward algorithms. Their main idea is to find contiguous repetitive sub-sequences and pre-compute all quantities of interest for these sub-sequences so that these quantities can be used multiple times without repeating the computation. Mozes *et. al.* have shown that, despite being one of the simplest compression techniques, the four Russians method yields a logarithmic improvement over the traditional Viterbi algorithm. That the four Russians method improves dynamic programming algorithms for other applications has been shown previously [13,20,22,23]. Morever, Mozes *et. al.* have shown that for an HMM with few states Baum-Welch training can be improved using partially computed forward and backward variables.

While [17,21] shows asymptotic speed up for the Viterbi algorithm and improved Baum-Welch training, we focus on Bayesian analysis of HMM using MCMC simulations. Following their idea we pre-compute quantities of interest for all possible $\log T$-sized sub-sequences (Note: in the following we assume sub-sequences to be contiguous) and use these quantities to compute $O\left(\frac{T}{\log T}\right)$ forward variables. While forward-backward Gibbs sampling needs T forward variables, we show that, because of the conditional dependency structure in an HMM, one can use the partially computed forward variables to implement a modified, but exact, version of forward-backward Gibbs sampling. As forward variable computations dominate the running time we achieve a $O(\log T)$ speed-up.

To demonstrate the effectiveness of our method on biological problems, we apply it to Bayesian analysis of DNA segmentation [4,5,6,18]. A *segment* is defined to be a contiguous region of DNA sequence, where nucleic acid composition is assumed to follow the same distribution. For example isochore classes can be identified by solving the DNA segmentation problem using HMMs [11]; see [4,5] for a fully Bayesian approach.

In summary, we

- utilize characteristics of the discrete-valued observation sequence to improve the running time of MCMC sampling. To the best of our knowledge this is the first use of sequence repetitions in improving Bayesian HMM computations,
- prove that sequence repetitions can be used to speed-up MCMC sampling by a factor of $\Theta(\log T)$. Note that the speed-up we achieve is as large as the one in [17,21] for forward variable computations, and
- experimentally verify that the theoretical speed-up is also observed in practical problems like detecting homogeneous segments in DNA, where we achieve a speed-up of up to 5 on bacterial genomes.

2 Hidden Markov Model

We consider HMM with discrete emission distributions; see [27] for an introduction. We will use the following notation: N denotes the number of states, $S = \{s_1, s_2, \ldots, s_N\} \equiv \{1, 2, \ldots, N\}$ the set of states, $\Sigma = \{1, \ldots, |\Sigma|\}$ finite alphabet, $O = (o_1, o_2, \ldots, o_T) \in \Sigma^T$ the observation sequence, $Q = (q_1, q_2, \ldots, q_T) \in S^T$ the hidden state sequence, $A = \{a_{i,j}\}_{1\leq i,j\leq N}$ the transition matrix, $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$ the initial distribution over states, γ the order of observation process, $o_t^{'} = (o_{\max(1,t-\gamma)}, \ldots, o_{t-1})$, and $B = \{b_{i,j}^{\beta}\}_{\beta\in[\Sigma\cup\Sigma^2\cup\ldots\cup\Sigma^\gamma],1\leq i\leq N,1\leq j\leq|\Sigma|}$ the emission matrix.

The hidden state sequence Q follows a first-order markov chain, that is

$$P(q_1) = \pi_{q_1} \text{ , and} \tag{1}$$

$$P(q_t|q_1, \ldots, q_{t-1}) = P(q_t|q_{t-1}) = a_{q_{t-1},q_t} \ . \tag{2}$$

In contrast to the usual literature, where emissions only depend on the state, we consider the case of higher order emissions [16]. Then the probability of an observation sequence O can be described using the following equation.

$$P(o_t|q_1, \ldots, q_t, o_1, \ldots, o_{t-1}) = P(o_t|q_t, o_{\max(1,t-\gamma)}, \ldots, o_{t-1}) = b_{q_t,o_t}^{o_t^{'}} \ . \tag{3}$$

Fig. 1 shows the dependency structure in HMM using graphical models for regular ($\gamma = 0$) and first-order ($\gamma = 1$) emission HMMs.

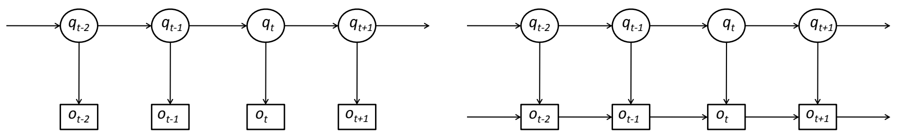


Fig. 1. Graphical model showing conditional dependency for HMM; $\gamma = 0$ (left) and $\gamma = 1$ (right). To simplify notation we also use q_t for the state and o_t for the emission random variables. An arrow from X to Y means Y is dependent on X.

3 Bayesian Analysis: Forward-Backward Gibbs Sampling

To use Bayesian analysis we need to choose prior distributions. In this work we use the Dirichlet distribution as prior for $A_{i,*}$, $B_{i,*}$, and π. Our analysis will still be valid for any standard conjugate prior distribution.

As we are interested in computing the distribution $P(Q|O)$, and closed form solutions of the Bayesian integral are not feasible, the use of MCMC techniques like Gibbs sampling or Metropolis-Hastings becomes mandatory [3,14]. MCMC algorithms create a Markov chain that has the desired distribution, here $P(Q|O)$, as its stationary distribution. After an appropriate burn-in, performing a random walk on the state transition graph, the state of the chain can be used as a sample from the stationary distribution [14]. Scott [29] compares various MCMC approaches and strongly argues in favor of forward-backward Gibbs sampling (FBG-sampling) for it's excellent convergence characteristics. Here we will restrict our discussion to FBG-sampling only. We define forward variables as $\alpha_t(j) = P(q_t = j, o_1, \ldots, o_t|A, B, \pi)$ and briefly summarize FBG-sampling for a HMM $\equiv (A, B, \pi) = \theta$ in Alg. 1; see [9,29] for details.

Algorithm 1. FBG-Sampling(O)

```
1: Choose initial parameters $\theta^0 = (A^0, B^0, \pi^0)$.
2: Perform the following steps for $0 \leq m < M$.
   (a) $Q^m$ = *StateSampler(O, $\theta^m$)* [See Alg. 2]
   (b) Sample HMM parameters,
       $\theta^{m+1} \sim PriorDistribution(hyperparameters, O, Q^m, \theta^m)$
3: **return** $Q^0, Q^1, \ldots, Q^{M-1}$.
```

FBG-sampling starts with an initial choice of parameters θ^0 and alternatively keeps sampling state sequence Q^m and parameters θ^{m+1}. See [9] for a proof that Q^m returned by Alg. 2 is indeed sampled from the marginal distribution $P(Q^m|O, \theta^m)$.

Algorithm *StateSampler* uses $O(TN^2)$ space and runs in $O(TN^2)$ time; step 1 (forward variables) runs in $O(TN^2)$ time and step 2 (backward sampling) in $O(T \log N)$. It is obvious from the above algorithm that all the pre-computed forward variables are not used for sampling the state sequence Q^m. In the next section we will see that even without computing all forward variables Q^m can be sampled accurately.

4 Speeding Up MCMC

We reformulate the forward variables α using matrix notation following [17,21]. Let $M^u(v)$, where $u \in [\Sigma \cup \Sigma^2 \cup \ldots \cup \Sigma^\gamma]$ and $v \in \Sigma$, be a $N \times N$ matrix with

Algorithm 2. StateSampler(O, θ)

```
1: Forward Variables:
     – Compute $\alpha_1(j) = P(o_1, q_1 = j|\theta) = \pi_j b_{j,o_1}^{o_1'}$ for all $j$.
     – For $2 \leq t \leq T$:
           Compute $\alpha_t(j) = P(o_1 o_2 \dots o_t, q_t = j|\theta) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{i,j} b_{j,o_t}^{o_t'}$ for all $j$.
2: Backward Sampling:
     – Sample $q_T$ s.t. $P(q_T = i) \propto \alpha_T(i)$.
     – For $T > t \geq 1$:
           Sample $q_t$ s.t. $P(q_t = i) \propto \alpha_t(i) a_{i,q_{t+1}}$.
3: return $Q$
```

elements $M_{i,j}^{u}(v) = a_{i,j} b_{j,v}^{u}$. Forward variables at time t, α_t, can be rewritten as a row vector,

$$\alpha_t = \pi \cdot M^{o_1'}(o_1) \cdot M^{o_2'}(o_2) \cdot \dots \cdot M^{o_{t-1}'}(o_{t-1}) \cdot M^{o_t'}(o_t) \tag{4}$$

$$= \alpha_{t-1} \cdot M^{o_t'}(o_t) \ . \tag{5}$$

It is important to note that the matrix formulation does not change the running time of the algorithm.

4.1 Compression and Forward Variables

Lets define $O_{i\dots j} := o_i o_{i+1} \dots o_j$ and $Q_{i\dots j} := q_i q_{i+1} \dots q_j$. We define the matrix $M^{o_i'}(O_{i\dots j})$ as

$$M^{o_i'}(O_{i\dots j}) = M^{o_i'}(o_i) \cdot M^{o_{i+1}'}(o_{i+1}) \cdot \dots \cdot M^{o_{j-1}'}(o_{j-1}) \cdot M^{o_j'}(o_j) \ . \tag{6}$$

We assume that the length of the observation sequence, T, is a multiple of k such that $d = \frac{T}{k}$ and create groups of fixed size from the observation sequence $O = O_{1\dots k} O_{k+1\dots 2k} \dots O_{(d-1)k+1\dots T}$. Pre-computing all possible matrices $M(X)$, where $|X| \leq k$, for future use is informally known as the *four Russians method.* Now α_{lk} can be expressed using (6) as

$$\alpha_{lk} = \pi \cdot M^{o_1'}(O_{1\dots k}) \cdot M^{o_{k+1}'}(O_{k+1\dots 2k}) \cdot \dots \cdot M^{o_{(l-1)k+1}'}(O_{(l-1)k+1\dots lk}) \tag{7}$$

$$= \alpha_{(l-1)k} \cdot M^{o_{(l-1)k+1}'}(O_{(l-1)k+1\dots lk}) \ . \tag{8}$$

The compressed sequence allows us to skip computing forward variables inside a group, which results in significant time savings. [17,21] similarly defines forward variables using compressed sequence to improve Baum-Welch training. Note that we cannot directly use the backward sampling in Alg. 2 in this setting. In the remaining part of this section we will explain how we can overcome this problem.

4.2 Backward-Forward State Sequence

Now we will modify the order of state sampling, turning *backward sampling* step of Alg. 2 into *backward-forward sampling*, and express the distribution $P(Q|O,\theta)$ in a way that helps us to sample Q accurately and efficiently. We write

$$P(Q|O,\theta) = \underbrace{P(Q_{1\ldots k-1}|Q_{k\ldots T},O,\theta)}_{\text{Part A}}\underbrace{P(Q_{k\ldots T}|O,\theta)}_{\text{Part B}} \ . \tag{9}$$

By repeated application of Bayes theorem we can show that part B is proportional to

$$\underbrace{P(q_T|O,\theta)}_{\text{Part } B_1} \prod_{\substack{d\geq i\geq 2\\ s=(i-1)k\\ e=ik}} \left(\underbrace{P(q_s|Q_{e\ldots T},O,\theta)}_{\text{Part } B_2} \prod_{j=s+1}^{e-1} \underbrace{P(q_j|Q_{s\ldots j-1},Q_{e\ldots T},O,\theta)}_{\text{Part } B_3} \right) . \tag{10}$$

Part B_1, B_2, and B_3 can be sampled using the following relations.

Sampling $\boldsymbol{B_1}$

$$\begin{aligned} P(q_T|O,\theta) &\propto P(q_T,O|\theta) \\ &\propto \alpha_T(q_T) \end{aligned} \tag{11}$$

Sampling $\boldsymbol{B_2}$

$$\begin{aligned}
&P(q_s|Q_{e\ldots T},O,\theta)\\
&= P(q_s|Q_{e\ldots T},O_{1\ldots s},O_{s+1\ldots T},\theta)\\
&\propto P(q_s|O_{1\ldots s},\theta)P(O_{s+1\ldots T},Q_{e\ldots T}|q_s,O_{1\ldots s},\theta) && (12)\\
&= P(q_s|O_{1\ldots s},\theta)P(O_{s+1\ldots T},Q_{e\ldots T}|q_s,o^{'}_{s+1},\theta) && (13)\\
&\propto P(q_s,O_{1\ldots s}|\theta)P(O_{s+1\ldots e},O_{e+1\ldots T},q_e,Q_{e+1\ldots T}|q_s,o^{'}_{s+1},\theta) && (14)\\
&= \alpha_s(q_s)P(O_{s+1\ldots e},q_e|q_s,o^{'}_{s+1},\theta)P(O_{e+1\ldots T},Q_{e+1\ldots T}|q_s,O_{s+1\ldots e},q_e,o^{'}_{s+1},\theta) && (15)\\
&= \alpha_s(q_s)P(O_{s+1\ldots e},q_e|q_s,o^{'}_{s+1},\theta)P(O_{e+1\ldots T},Q_{e+1\ldots T}|O_{s+1\ldots e},q_e,o^{'}_{s+1},\theta) && (16)\\
&\propto \alpha_s(q_s)P(O_{s+1\ldots e},q_e|q_s,o^{'}_{s+1},\theta) && (17)\\
&= \alpha_s(q_s)M^{o^{'}_{s+1}}_{q_s,q_e}(O_{s+1\ldots e}) && (18)
\end{aligned}$$

Equation (12), (14), and (15) are derived from Bayes theorem. The conditional dependency structure of the HMM given $Q_{e\ldots T}$ (see Fig. 2) is used in (13) and (16). As the last term in (16) is independent of q_s it is dropped in (17).

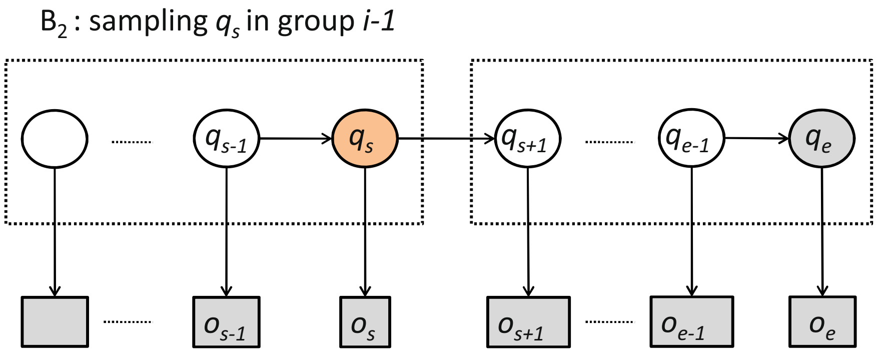


Fig. 2. Conditional dependency shown for sampling q_s using B_2 for $\gamma = 0$. Lightly shaded variables are either observed or already sampled. Dashed rectangle represents a group of observations. Here q_t, o_t are used as the random variable for state and emission to simplify notation.

Sampling B_3

$$
\begin{aligned}
&P(q_j|Q_{s\ldots j-1}, Q_{e\ldots T}, O, \theta)\\
&\propto P(q_j, o_j, Q_{e\ldots T}, O_{j+1\ldots T}|Q_{s\ldots j-1}, O_{1\ldots j-1}, \theta)\\
&= P(q_j, o_j|Q_{s\ldots j-1}, O_{1\ldots j-1}, \theta)P(Q_{e\ldots T}, O_{j+1\ldots T}|Q_{s\ldots j}, O_{1\ldots j}, \theta) \qquad (19)\\
&= P(q_j, o_j|q_{j-1}, o_j^{'}, \theta)P(Q_{e\ldots T}, O_{j+1\ldots T}|q_j, o_{j+1}^{'}, \theta) \qquad (20)\\
&= P(q_j, o_j|q_{j-1}, o_j^{'}, \theta)P(q_e, O_{j+1\ldots e}, Q_{e+1\ldots T}, O_{e+1\ldots T}|q_j, o_{j+1}^{'}, \theta)\\
&= P(q_j, o_j|q_{j-1}, o_j^{'}, \theta)P(q_e, O_{j+1\ldots e}|q_j, o_{j+1}^{'}, \theta)P(Q_{e+1\ldots T}, O_{e+1\ldots T}|q_e, O_{j+1\ldots e}, q_j, o_{j+1}^{'}, \theta) \qquad (21)\\
&= P(q_j, o_j|q_{j-1}, o_j^{'}, \theta)P(q_e, O_{j+1\ldots e}|q_j, o_{j+1}^{'}, \theta)P(Q_{e+1\ldots T}, O_{e+1\ldots T}|q_e, o_{j+1}^{'}, \theta) \qquad (22)\\
&\propto P(q_j, o_j|q_{j-1}, o_j^{'}, \theta)P(q_e, O_{j+1\ldots e}|q_j, o_{j+1}^{'}, \theta) \qquad (23)\\
&= M^{o_j^{'}}_{q_{j-1},q_j}(o_j)M^{o_{j+1}^{'}}_{q_j,q_e}(O_{j+1\ldots e}) \qquad (24)
\end{aligned}
$$

Equation (19) and (21) are derived from Bayes theorem. The conditional dependency structure of the HMM given $Q_{s\ldots j-1}$ and $Q_{e\ldots T}$ (see Fig. 3) is used in (20) and (22). As the last term in (22) is independent of q_j it is dropped in (23).

4.3 Fast Sampling Algorithm

Now we formally describe the algorithm *FastStateSampler* (see Alg. 3) and analyze it's running time. Instead of using Alg. 2 (*StateSampler*) in step *2.a* of Alg. 1 (*FBG-sampling*) now we can use Alg. 3 for fast MCMC simulations.

In the *Precompute* step of Alg. 3, $M^{\beta}(X)$ matrices, which are required in (18) and (24), are computed at first. To sample q_j using (24) (in *Backward-forward Sampling* step) we need to compute $M^{o_j^{'}}_{q_{j-1},q_j}(o_j)M^{o_{j+1}^{'}}_{q_j,q_e}(O_{j+1\ldots e})$ for all possible values of q_j, which is an $O(N)$ operation. Considering these quantities as weights

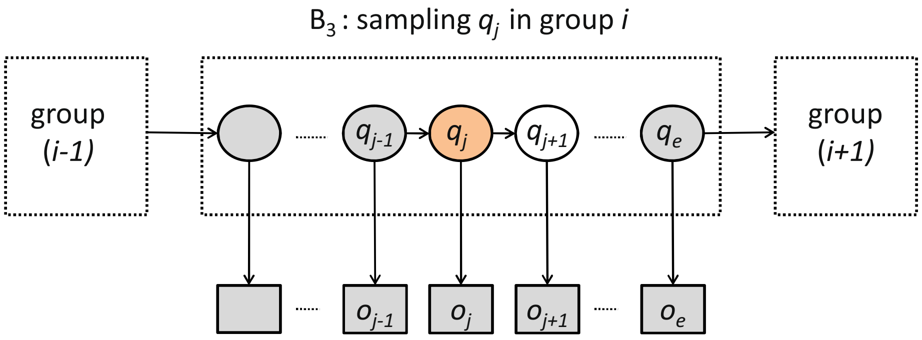


Fig. 3. Conditional dependency shown for sampling q_j using B_3 for $\gamma = 0$. Lightly shaded variables are either observed or already sampled. Dashed rectangle represents a group of observations. Here q_t, o_t are again used as the random variable for state and emission to simplify notation.

for possible states we can select q_j using weighted random sampling, which again takes $O(N)$ time. Interestingly, these quantities are already precomputed as intermediate parts of $M^{\beta}(O_{j\ldots e})$. Instead of simply storing these weights, if we store the sum of these values from state 1 to c in $R^{\beta}_{q_{j-1},q_e,c}(o_j, O_{j+1\ldots e})$, we can use binary search to select q_j in $O(\log N)$ time. Similarly, we store the sum of intermediate parts of α_* in $\delta_{*,*,*}$ to sample q_s using binary search.

Running Time. As there are at most $2|\Sigma|^{k+\gamma}$ matrices to be precomputed, the pre-computation step takes $O(2|\Sigma|^{k+\gamma}N^3)$ time. Forward variables are computed in $O(\frac{T}{k}N^2)$ time. Using the stored values in R and δ, the state sequence is sampled in $O(T\log N)$ time (the small portion where Alg. 2 is used does not affect the order of the algorithm). The total running time is $O(2|\Sigma|^{k+\gamma}N^3 + \frac{T}{k}N^2 + T\log N)$. If k is chosen to be $\frac{1}{2}\log_{|\Sigma|}T - \gamma$, the total running time becomes $O(2\sqrt{T}N^3 + \frac{2TN^2}{\log_{|\Sigma|}T-\gamma} + T\log N)$. Assuming $N < \frac{\sqrt{T}}{\log_{|\Sigma|}T-\gamma}$, *FastStateSampler* achieves a speed-up of $\Theta(\log_{|\Sigma|}T - \gamma)$ and uses $O(\frac{T}{\log_{|\Sigma|}T-\gamma}N^2)$ space.

5 Empirical Results

In this section we apply our fast sampling technique to a Bayesian analysis of DNA segmentation. A segment is defined as a contiguous region of a DNA sequence with similar nucleic acid composition. Many DNA sequences can be divided into homogeneous segments and interesting structures such as isochores can be extracted from the segmentation. We compare the performance of our method on the DNA segmentation problem with standard FBG-sampling.

We use Dirichlet priors and non-informative hyper-parameters as model parameter distributions. We measure the running time of forward-backward Gibbs sampling (Alg. 1) using both Alg. 2 and Alg. 3 as the sampler in step *2.a*. The running time of forward-backward Gibbs sampling is proportional to the

Algorithm 3. FastStateSampler(O, θ)

```
1: Precompute:
    – $M^{\beta}(X)$ for all $X \in \cup_{i=1}^{k} \Sigma^i$ and $\beta \in \cup_{i=1}^{\gamma} \Sigma^i$.
    – $R^{\beta}(x, X)$ for all $\beta \in \cup_{i=1}^{\gamma} \Sigma^i$, $x \in \Sigma$, and $X \in \cup_{i=1}^{k-1} \Sigma^i$ such that
        • $R^{\beta}_{i,j,1}(x, X) = M^{\beta}_{i,1}(x) M^{(\beta_{2 \ldots |\beta|}, x)}_{1,j}(X)$.
        • $R^{\beta}_{i,j,c}(x, X) = R^{\beta}_{i,j,c-1}(x, X) + M^{\beta}_{i,c}(x) M^{(\beta_{2 \ldots |\beta|}, x)}_{c,j}(X)$ for $1 < c \leq N$.
2: Forward Variables:
    – Compute $\alpha_k = \pi M^{0'_1}(O_{1 \ldots k})$.
    – For $1 < i \leq m$ and $1 \leq j \leq N$, compute $\alpha_{ik}$ and $\delta_{ik,j,*}$ in the following way.
        • $\delta_{ik,j,1} = \alpha_{(i-1)k}(1) M^{o'_{(i-1)k+1}}_{1,j}(O_{(i-1)k+1 \ldots e})$.
        • $\delta_{ik,j,c} = \delta_{ik,j,c-1} + \alpha_{(i-1)k}(c) M^{o'_{(i-1)k+1}}_{c,j}(O_{(i-1)k+1 \ldots e})$ for $1 < c \leq N$.
        • Set $\alpha_{ik}(j) = \delta_{ik,j,N}$.
3: Backward-forward Sampling:
    – Sample $q_T$ from (11).
    – For $m \geq i \geq 2$:
        • Let $s = (i-1)k$ and $e = ik$.
        • Sample $q_s$ from (18) by applying binary search on the monotonically increasing sequence $\delta_{s,q_e,1}, \delta_{s,q_e,2}, \ldots, \delta_{s,q_e,N}$.
        • For $s < j < e$, sample $q_j$ from (24) by applying binary search on the monotonically increasing sequence $R^{o'_j}_{q_{j-1},q_e,1}(o_j, O_{j+1 \ldots e})$, $R^{o'_j}_{q_{j-1},q_e,2}(o_j, O_{j+1 \ldots e})$, $\ldots$, $R^{o'_j}_{q_{j-1},q_e,N}(o_j, O_{j+1 \ldots e})$.
    – Given $q_k$, sample $q_1, q_2, \ldots, q_{k-1}$ (part $A$) using a slightly modified version of Alg. 2.
4: return $Q$
```

number of sampling iterations M (see step *2* of Alg. 1). We set $M = 10$ and compare execution time of one run of the algorithms. In [4] Boys *et. al.* used $500,000$ iterations to segment *Bacteriophage lambda* DNA. They showed that $6 \leq N \leq 8$ and $0 \leq \gamma \leq 2$ produced the best segmentation for *Bacteriophage lambda*. Unlike their model we keep γ and N fixed, but it can easily be modified to variable model dimensions. Four bacterial genomes — *Bacteriophage lambda* (genome size 0.05 Mbp), *Mycoplasma leachii* (1 Mbp), *Planctomyces brasiliensis* (6 Mbp), and *Sorangium cellulosum* (13 Mbp) — are segmented and the running time for different choices of N and γ are shown in Fig. 4. As both algorithms converge to the same stationary distribution we do not report any segmentation error.

As expected, we see logarithmic speed-up for our method over standard FBG-sampling (see Table 1). As the size of the dataset increases, so does the speed-up we observe. For *Sorangium cellulosum* we achieve a speed-up of 5. For small values of N, the state path sampling time is comparable to the pre-computation and forward variable computation time. As a result there is no significant speed-up for small N. For very large $N > \frac{\sqrt{T}}{\log_{|\Sigma|} T - \gamma}$ (often impractical) the algorithm gradually loses it's advantage. over standard FBG-sampling. However, this bound and overall running time can be improved by computing $M^{\beta}(X)$ matrices using fast matrix multiplication of order $o(N^3)$.

We implemented the algorithms in C++ and tested in a Linux machine with a 2.2 GHz AMD Opteron processor. As there was very little variation between the running time of two different runs of an algorithm, instead of averaging over multiple runs, we report the running time of one single run in Fig. 4.

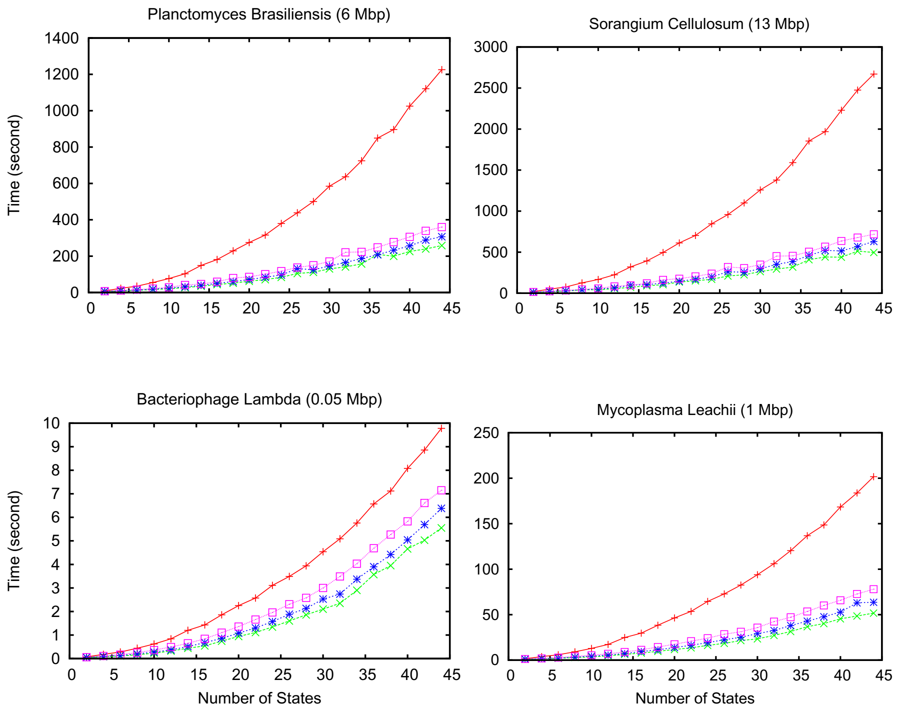


Fig. 4. Running time comparison on four datasets. Execution times for forward-backward Gibbs (red, $+$) and four Russians method ($\gamma = 0$ with (green, $\times$), $\gamma = 1$ with (blue, $*$), and $\gamma = 2$ with (pink, $\square$)) are shown.

Table 1. Speed-up using fast sampling method for HMM with $\gamma = 0, 1, 2$

Dataset	HMM Order (γ)	Number of States (N)										
		4	8	12	16	20	24	28	32	36	40	44
Bacteriophage lambda (0.05 Mbp)	0	2.2	2.8	2.6	2.6	2.4	2.3	2.1	2.2	1.8	1.7	1.8
	1	2.0	2.2	2.3	2.2	2.1	2.0	1.9	1.9	1.7	1.6	1.5
	2	1.8	1.8	1.8	1.8	1.6	1.6	1.6	1.5	1.4	1.3	1.3
Mycoplasma leachii (1 Mbp)	0	2.6	3.2	3.6	3.8	4.0	4.0	3.8	3.9	3.7	3.7	3.9
	1	2.4	2.7	3.1	3.2	3.4	3.3	3.3	3.3	3.2	3.2	3.1
	2	2.2	2.3	2.6	2.7	2.6	2.7	2.8	2.6	2.5	2.4	2.5
Planctomyces brasiliensis (6 Mbp)	0	2.6	3.4	3.8	4.2	4.4	4.5	4.5	4.5	4.1	4.5	4.8
	1	2.4	2.9	3.3	3.6	3.9	4.0	4.0	3.9	4.0	4.0	3.8
	2	2.3	2.5	2.8	3.1	3.1	3.3	3.4	3.0	3.2	3.2	3.3
Sorangium cellulosum (13 Mbp)	0	2.7	3.0	4.1	4.4	4.5	5.0	4.9	4.7	4.6	5.1	5.4
	1	2.5	3.0	3.5	3.8	4.2	4.3	4.3	4.0	4.0	4.3	4.1
	2	2.3	2.5	2.9	3.3	3.4	3.6	3.8	3.2	3.5	3.4	3.7

6 Conclusion

In this paper we have presented a modified version of the forward-backward Gibbs sampling algorithm for Bayesian analysis with a logarithmic improvement in running time. We have used the four Russians method to pre-compute all possible quantities of future interest and shown that exact sampling can work with fewer forward variables by using the pre-computed quantities. To the best of our knowledge, this is the first use of sequence repetition in discrete sequences for faster MCMC simulations.

As biological sequences are often long and the alphabet size is small, our approach can be adopted to make Bayesian computations faster in biological applications. We have demonstrated the advantage of our method on the DNA segmentation problem where we have achieved speed-ups similar to other applications of four Russians method.

A natural extension to our approach would be applying other compression schemes. In some cases, when observations in a sequence are less uniform in nature, other schemes may outperform the four Russians method. It will also be interesting to apply our method, by taking advantage of faster computations, to the problems where Bayesian analysis was not favored previously.

References

1. Andrec, M., Levy, R.M., Talaga, D.S.: Direct determination of kinetic rates from single-molecule photon arrival trajectories using Hidden Markov Models. The Journal of Physical Chemistry A 107(38), 7454–7464 (2003), PMID: 19626138
2. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. The Annals of Mathematical Statistics 41(1), 164–171 (1970)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer-Verlag New York, Inc., Secaucus (2006)

4. Boys, R.J., Henderson, D.A.: A Bayesian approach to DNA sequence segmentation. Biometrics 60(3), 573–581 (2004)
5. Boys, R.J., Henderson, D.A., Wilkinson, D.J.: Detecting homogeneous segments in DNA sequences by using Hidden Markov Models. Journal of the Royal Statistical Society. Series C (Applied Statistics) 49(2), 269–285 (2000)
6. Braun, J.V., Muller, H.-G.: Statistical methods for DNA sequence segmentation. Statistical Science 13(2), 142–162 (1998)
7. Buchsbaum, A.L., Giancarlo, R.: Algorithmic aspects in speech recognition: an introduction. J. Exp. Algorithmics 2 (January 1997)
8. Burge, C., Karlin, S.: Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology 268(1), 78–94 (1997)
9. Chib, S.: Calculating posterior distributions and modal estimates in Markov mixture models. Journal of Econometrics 75(1), 79–97 (1996)
10. Churchill, G.: Stochastic models for heterogeneous DNA sequences. Bulletin of Mathematical Biology 51, 79–94 (1989), doi:10.1007/BF02458837
11. Churchill, G.A.: Hidden Markov chains and the analysis of genome structure. Computers and Chemistry 16(2), 107–115 (1992)
12. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.J.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge (1998)
13. Frid, Y., Gusfield, D.: A simple, practical and complete $O(n\hat{3}/\log n)$-time algorithm for RNA folding using the Four-Russians speedup. In: Salzberg, S.L., Warnow, T. (eds.) WABI 2009. LNCS, vol. 5724, pp. 97–107. Springer, Heidelberg (2009)
14. Gilks, W., Gilks, W., Richardson, S., Spiegelhalter, D.: Markov chain Monte Carlo in practice. Interdisciplinary statistics. Chapman & Hall, Boca Raton (1996)
15. Guha, S., Li, Y., Neuberg, D.: Bayesian Hidden Markov Modeling of Array CGH data. Journal of the American Statistical Association 103, 485–497 (2008)
16. Krogh, A.: Two methods for improving performance of a HMM and their application for gene finding. In: Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology, pp. 179–186. AAAI Press, Menlo Park (1997)
17. Lifshits, Y., Mozes, S., Weimann, O., Ziv-Ukelson, M.: Speeding up HMM decoding and training by exploiting sequence repetitions. Algorithmica 54(3), 379–399 (2009)
18. Liu, J.S., Lawrence, C.E.: Bayesian inference on biopolymer models. Bioinformatics 15(1), 38–52 (1999)
19. Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT Press, Cambridge (1999)
20. Masek, W.J., Paterson, M.S.: A faster algorithm computing string edit distances. Journal of Computer and System Sciences 20(1), 18–31 (1980)
21. Mozes, S., Weimann, O., Ziv-Ukelson, M.: Speeding up HMM decoding and training by exploiting sequence repetitions. In: Ma, B., Zhang, K. (eds.) CPM 2007. LNCS, vol. 4580, pp. 4–15. Springer, Heidelberg (2007)
22. Myers, E.: An O(ND) difference algorithm and its variations. Algorithmica 1, 251–266 (1986), doi:10.1007/BF01840446
23. Myers, G.: A Four Russians algorithm for regular expression pattern matching. J. ACM 39, 432–448 (1992)
24. Nicolas, P., Bize, L., Muri, F., Hoebeke, M., Rodolphe, F., Ehrlich, S.D., Prum, B., Bessires, P.: Mining bacillus subtilis chromosome heterogeneities using Hidden Markov Models. Nucleic Acids Research 30(6), 1418–1426 (2002)

25. Ohler, U., Niemann, H.: Joint modeling of DNA sequence and physical properties to improve eukaryotic promoter recognition. Bioinformatics 17(suppl. 1), S199–S206 (2001)
26. Patterson, T.A., Thomas, L., Wilcox, C., Ovaskainen, O., Matthiopoulos, J.: State-space models of individual animal movement. Trends in Ecology and Evolution 23(2), 87–94 (2008)
27. Rabiner, L.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
28. Redelings, B.D., Suchard, M.A.: Joint Bayesian estimation of alignment and phylogeny. Systematic Biology 54(3), 401–418 (2005)
29. Scott, S.: Bayesian methods for Hidden Markov Models: Recursive computing in the 21st century. Journal of the American Statistical Association, 337–351 (March 2002)
30. Scott, S.L.: A Bayesian paradigm for designing intrusion detection systems. Computational Statistics and Data Analysis 45(1), 69–83 (2004); Computer Security and Statistics
31. Sims, C.A., Zha, T.: Were there regime switches in U.S. monetary policy? The American Economic Review 96(1), 54–81 (2006)

Using Dominances for Solving the Protein Family Identification Problem

Noel Malod-Dognin[1], Mathilde Le Boudic-Jamin[2], Pritish Kamath[3], and Rumen Andonov[2,⋆]

[1] INRIA Sophia Antipolis - Méditerranée, France
[2] INRIA Rennes - Bretagne Atlantique and University of Rennes 1, France
[3] Computer Science and Engineering Department, Indian Institute of Technology, India
{Noel.Malod-Dognin,mathilde.le_boudic-jamin}@inria.fr, pritish.kamath@iitb.ac.in, randonov@irisa.fr

Abstract. Identification of protein families is a computational biology challenge that needs efficient and reliable methods. Here we introduce the concept of dominance and propose a novel combined approach based on Distance Alignment Search Tool (DAST), which contains an exact algorithm with bounds. Our experiments show that this method successfully finds the most similar proteins in a set without solving all instances.

Keywords: Protein structure comparison, classification, bounds, dominance.

1 Introduction

The 3D structure of macro-molecules underpins all biological functions. Similarities between protein structures may come from evolutionary relationships [1,2], and similar protein structures relate to similar functions. Thus, the protein structure comparison is a key tool in structural biology, whose primary goal is to understand function through structure. During the last decades, many protein structure comparison approaches have been proposed, each aiming at quantifying the intuitive notion of structural similarity. Most of the proposed methods are either based on optimal rigid-body superimposition (like VAST[3] or STRUCTAL[4]), whose computation is based on the least Root Mean Square Deviation of residue coordinates (*RMSDc*) as first defined by Kabsch[5], or on the comparison of the internal distances between the residues (like DALI[6], CMO[7] or DAST[8]). The main challenge in protein structure comparison is to design efficient algorithms, since the comparison of two protein structures is often NP-complete, as first shown in [9].

1.1 DAST

DAST (Distance-based Alignment Search Tool) is a protein structure comparison method based on internal distances [8]. In DAST, two proteins p_1 and p_2 are represented by their ordered sets of residues N_1 and N_2. The matching between residues

⋆ Corresponding author.

$i \in N_1$ and $k \in N_2$, denoted by $i \leftrightarrow k$, is allowed only if i and k come from the same kind of secondary structure (i.e. if either i and k both come from α-helices, or both come from β-strands, or both come from loops). By assumption, for any pair of residues i and j from the same protein we know the euclidean distance between their α-carbons (denoted here by d_{ij}).

Definition 1. *Pair (i, j) from p_1 is compatible with pair (k,l) from p_2 if and only if: (1) $i < j$ and $k < l$ (order preserving) and ; (2) $|d_{ij} - d_{kl}| \leq \tau$, where τ is a given distance threshold (isometric relationship).*

If (i, j) is compatible with (k,l) we are allowed to match (align) $i \leftrightarrow k$ and $j \leftrightarrow l$ (i.e. they form a matching pair). An optimal structural alignment is given by the longest sequence of matching pairs "$i_1 \leftrightarrow k_1$, $i_2 \leftrightarrow k_2$, ..., $i_t \leftrightarrow k_t$" in which any two matching pairs are compatible. As shown in section 3.2, DAST is equivalent to solving a maximum clique problem and is thus is an NP-Complete problem [10]. Set *ncr* (number of common residues) to denote the length of the optimal alignment between p_1 and p_2. Two similarity scores can be used:

$$S^G_{DAST}(p_1, p_2) = \frac{2 \times ncr}{|N_1| + |N_2|} \quad \text{and} \quad S^L_{DAST}(p_1, p_2) = \frac{ncr}{\min(|N_1|, |N_2|)}. \tag{1}$$

The first score, $S^G_{DAST}(p_1, p_2)$, is normalized according to the mean of two proteins and is oriented to detect global similarity between p_1 and p_2. The second one, $S^L_{DAST}(p_1, p_2)$, is normalized according to the smallest between the two proteins and is more suitable to detect local similarity.

Conceptually, DAST is inspired from the CMO approach which has been largely studied in the literature [7,11,12]. Both approaches aim to maximize the number of compatible matching pairs. However, the CMO compatibility does not consider the isometric condition from definition 1. As a consequence, the underlying optimization problems, the behavior of the corresponding solvers, as well as the characteristics of the provided alignments differ in both approaches. The most interesting feature of DAST is that it always returns an alignment (matching) of good quality, i.e. having a Root Mean Squared Deviation of internal distances ($RMSD_d$) which is less or equal than the given threshold τ (see [8]). This property is not guaranteed in CMO. On the other hand, a very efficient exact algorithm for finding the longest augmented path (the optimization problems in CMO) has been recently proposed in [12]. The associate solver, A_purva, is significantly faster than the nowadays maximum cardinality clique solvers. These issues will be discussed and illustrated in the computational results section.

1.2 The Protein Family Identification Problem

The exponential growth of the number of known protein structures in the Protein Data Bank [13] over the past decade led to the problem of protein classification. We mean here how to automatically insert new protein structures into an already existing classified database such as CATH[14] or SCOP[15]. The problem of determining in which classes new structures belong, referred here as the Protein Family Identification Problem , can be defined as follows.

Definition 2. *Given a set of to-be-classified query protein structures $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$, a set of classified target protein structures $\mathcal{P}=\{p_1, p_2, \ldots, p_n\}$, and a protein structure similarity function $S : \mathcal{Q} \times \mathcal{P} \rightarrow \mathcal{R}^+$, the* ***Protein Family Identification Problem (FIP)*** *consists in classifying each query structure $q_i \in \mathcal{Q}$ in the class of it's nearest neighbor NN_i which is defined as $NN_i = \underset{p_j \in \mathcal{P}}{\operatorname{argmax}} S(q_i, p_j)$.*

There are computational pitfalls in the FIP . The number of similarity scores $S(q_i, p_j)$ that need to be computed is $|\mathcal{Q}| \times |\mathcal{P}|$, where $|\mathcal{P}|$ can be very large (there are currently 152920 classified protein structures in the expert classification CATH). Moreover, computing a single similarity score is often equivalent to solving a NP-hard problem (ex: DALI, DAST, CMO, VAST, etc...). Depending on how these NP-hard problems are solved, two cases are possible. First, if the solver is a heuristic (ex: DALI, VAST), then the similarity scores are only approximated, and thus the resulting classification is not optimal (according to the similarity function). Second, if the solver is exact (ex CMO, DAST), and because of the NP-hardness of the problem, some instances cannot be optimally solved within reasonable time, leading to either sub-optimal or to missing similarity scores, both implying that the obtained classification is not optimal, or cannot be computed if too many similarity scores are missing.

In this paper, we propose a notion of dominance between the protein structure comparison instances that allows the computation of optimal protein structure classifications without optimally solving all the comparison instances, and thus reduces the effect of the NP-Hardness of the similarity score. As presented in section 2, using dominance supposes to compute both upper and lower bound on the similarity score. In section 3.3, we propose an efficient bounding strategy for DAST.

2 Dominance

The idea behind the dominance is that in FIP problem, given a query protein structure $q \in \mathcal{Q}$ and a classified set of target protein structures $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, we are interested only in finding the nearest neighbor of q in $\mathcal{P}$, i.e. $NN_q = \underset{p_j \in \mathcal{P}}{\operatorname{argmax}} S(q, p_j)$. If a protein structure p_j can be proved not to be NN_q before $S(q, p_j)$ is optimally computed, then spending more time on proceeding $S(q, p_j)$ is useless.

Let us suppose that the solving process can be stopped with a time limit t, and can then return both a lower-bound $S_{min}(q, p_i)$ and an upper-bound $S_{max}(q, p_i)$ on the similarity score $S(q, p_i)$, i.e $S_{min}(q, p_i) \leq S(q, p_i) \leq S_{max}(q, p_i)$ (if the instance is optimally solved, then $S_{min}(q, p_i) = S(q, p_i) = S_{max}(q, p_i)$).

Definition 3. *Given a query $q \in \mathcal{Q}$ and two target protein structures $p_1 \in \mathcal{P}$ and $p_2 \in \mathcal{P}$, the instance (q, p_2)* ***dominates*** *the instance (q, p_1) if $S_{min}(q, p_2) \geq S_{max}(q, p_1)$.*

If the instance (q, p_2) dominates the instance (q, p_1), then $S(q, p_2) \geq S(q, p_1)$. Thus, p_1 is not the nearest neighbor of q and there is no need to continue computing $S(q, p_1)$. Moreover, if one instance (q, p_i) dominates all the other instances (q, p_j), $p_j \in \mathcal{P}$, then $NN_q = p_i$, and the entire procedure can be stopped (including the instance (q, p_i)).

From now on, we use the dominances to fasten the FIP as follows.

1. All instances (q_i, p_j), $q_i \in \mathcal{Q}$, $p_j \in \mathcal{P}$ are put in a queue, and a time limit argument t is set to a small value.
2. For each instances (q_i, p_j) in the queue, the similarity $S(q_i, p_j)$ is evaluated (by computing $S_{min}(q_i, p_j)$ and $S_{max}(q_i, p_j)$) within the time limit t.
3. All dominated instances are removed from the queue. If an instance (q_i, p_j) dominates all the other instances (q_i, p_k), $p_k \in \mathcal{P}$, then the nearest neighbor of q_i is set to p_j, and the instance (q_i, p_j) is also removed from the queue.
4. If the queue is empty, then the nearest neighbors of all the queries have been found and the FIP is optimally solved. Otherwise, the time limit t is increased, and steps 2 to 4 are repeated until the queue is empty.

3 Modifying DAST for Using Dominance

Using dominances supposes that the solution process can return upper and lower-bounds on the similarity score. Unfortunately, as presented in [8], DAST does not possess such features. This section presents how these bounds were added into DAST. First, in sections 3.2, we briefly recall DAST principle. Then, in section 3.3, we present our bounding strategy.

3.1 Notation and Definitions

Let us first introduce some notations and definitions coming from [12] and [8].

Definition 4. *A $m \times n$ **alignment graph** $G = (V, E)$ is a graph in which the vertex set V is depicted by a (m-rows) $\times$ (n-columns) array T, where each cell $T[i][k]$ contains at most one vertex $i.k$ from V (note that for both arrays and vertices, the first index stands for the row number, and the second for the column number). Two vertices $i.k$ and $j.l$ can be connected by an edge $(i.k, j.l) \in E$ only if $i < j$ and $k < l$. An example of such alignment graph is given in the figure 1:**Right**.*

Definition 5. *Given graph $G = (V, E)$, a **clique** is a subset S of V such that for any two vertices $u \in S$ and $v \in S$, $u \neq v$, u and v are connected by an edge (u, v) in E.*

Definition 6. *The **maximum clique problem** consists in finding in a graph $G = V, E$ a largest (in terms of vertices) clique, denoted by $MCC(G)$. The maximum clique problem is one of the first shown to be NP-complete [10].*

In a $n \times m$ alignment graph $G = (V, E)$, the subset of V restricted to the vertices in the rows $j > i$ and in the columns $l > k$ is denoted by $V^{i.k}$. Similarly, $\hat{V}^{i.k}$ is the subset of V restricted to the vertices in the rows j, $0 \leq j \leq i$ and in the columns l, $0 \leq l \leq k$. The subgraph of G induced by the vertices in $V^{i.k}$ is denoted by $G^{i.k}$, and the subgraph of G induced by the vertices in $\hat{V}^{i.k}$ is denoted by $\hat{G}^{i.k}$. The vertex $j.l$ is a successor of the vertex $i.k$ if $i < j$, $k < l$ and edge $(i.k, j.l)$ is in E, and the set of successors of a vertex $i.k$ is denoted by $\Gamma^+(i.k)$. The vertex $a.b$ is a predecessor of the vertex $i.k$ if $a < i$, $b < k$ and edge $(a.b, i.k)$ is in E, and the set of predecessor of a vertex $i.k$ is denoted by $\Gamma^-(i.k)$. The maximum clique in a graph G is denoted by $MCC(G)$, and its cardinality is denoted by $|MCC(G)|$. An upper-bound on $|MCC(G)|$ is denoted by $|\overline{MCC}(G)|$.

Definition 7. *An* ***increasing subset of vertices*** *in an alignment graph $G = \{V,E\}$ is an ordered subset $\{i_1.k_1, i_2.k_2, \dots, i_t.k_t\}$ of V, such that $\forall j \in [1,t-1]$, $i_j < i_{j+1}$, $k_j < k_{j+1}$. $LIS(G)$ is the longest, in terms of vertices, increasing subset of vertices of G.*

Definition 8. *An* ***increasing path*** *in an alignment $G = \{V,E\}$ is an increasing subset of vertex $\{i_1.k_1, i_2.k_2, \dots, i_t.k_t\}$ such that $\forall j \in [1,t-1]$, $(i_j.k_j, i_{j+1}.k_{j+1}) \in E$. The longest, in terms of vertices, increasing path in G is denoted by $LIP(G)$.*

Lemma 1. $|MCC(G)| \leq |LIP(G)| \leq |LIS(G)|$.

Proof. Since any two vertices in a clique are adjacent, definition 4 implies that a clique in G is both an increasing subset of vertices and an increasing path, thus $|MCC(G)| \leq |LIP(G)|$. Moreover, $LIP(G)$ is by definition an increasing subset of vertices, which implies that $|LIP(G)| \leq |LIS(G)|$.

3.2 Maximum Clique Formulation of DAST

DAST is rephrased as a maximum clique problem in an alignment graph as follows. Let G be a $|N_1| \times |N_2|$ alignment graph, where each row corresponds to a residue of N_1 and each column corresponds to a residue of N_2. A vertex $i.k$ is in V only if residues $i \in N_1$ and $k \in N_2$ both come from the same kind of secondary structure (i.e. if matching $i \leftrightarrow k$ is possible). An edge $(i.k, j.l)$ is in E if and only if (i) $i < j$ and $k < l$, for order preserving, and (ii) if $|d_{ij} - d_{kl}| \leq \tau$.

As illustrated in figure 1, an optimal matching between two protein structures p_1 and p_2 corresponds to a maximum clique in G. For example the maximum clique $\{(1.1),(2.2),(3.3)\}$ in figure 1:Right corresponds to the optimal matching between residues $(1,2,3)$ from p_1 and residues $(1,2,3)$ from p_2.

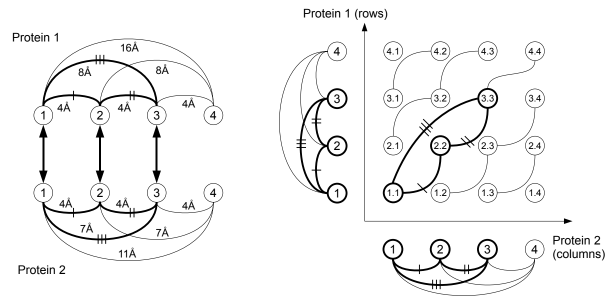


Fig. 1. Left: An optimal matching (represented by the arrows) between protein 1 and 2, when using a distance threshold of 1Å. **Right:** the corresponding maximum clique in the $|N_1| \times |N_2|$ alignment graph.

3.3 Bounding Strategy

Both DAST similarity scores [1] use *ncr*, the number of common residues, which is equal to $|MCC(G)|$. Thus, bounding DAST score is equivalent to provide a lower and upper-bound on the cardinality of the maximum clique in G, when the time limit t is reached.

The case of the lower-bound is trivial, since the best clique found so far, *Best*, is by definition a lower-bound of *ncr*. Computing an efficient upper-bound on $|MCC(G)|$ is less straightforward.

Intermediate state of execution. As explained in [8], the maximum clique solver of DAST visits the vertices of V in decreasing order of column (first) and decreasing order of row (second). For each visited vertex $i.k$, the clique solver computes (or upper-bounds) the maximum clique in $G^{i.k}$. If a clique larger than the current best one (*Best*) is found, then *Best* is updated. Finally, an array C is used to store, in each entry $C[i][k]$, the upper-estimated size of the maximum clique in $G^{i.k}$. This array is later used to fasten the maximum clique computation starting from a vertex having lower row and column indexes, and this implies that computation of $C[i][k]$ requires that $C[i+1][j]$, $C[i+1][j+1]$ and $C[i][j+1]$ are already computed. Since the evaluation of a given cell $T[i][k]$ (i.e. the computation of $C[i][k]$) only occurs after $C[i+1][j]$, $C[i+1][j+1]$ and $C[i][j+1]$ have been computed, an intermediate state of execution of DAST can be represented as in figure 2. In such intermediate state, the cells of T can be split in the following way.

- Cells in which $C[i][k]$ has already been evaluated will be referred to as *evaluated cells*. They correspond to the horizontally striped area in Figure 2:Left. The current best clique, *Best*, has been found in this set.
- Cells for which $C[i][k]$ have not yet been evaluated will be referred to as *unevaluated cells*.
- Unevaluated cells which are adjacent to an evaluated cell (either side-wise or diagonally) will be referred to as *boundary cells*, and are shown in black in Figure 2:Left. The unevaluated cells for which $C[i][k]$ can be computed (i.e. for which $C[i+1][j]$, $C[i+1][j+1]$ and $C[i][j+1]$ are already computed), belong by definition to the subset of the boundary cells, and are marked with a white square in Figure 2:Left. Finally, unevaluated cells that are not boundary cells are shown as vertically striped.

Upper bounding strategy. It is important to remember that in an intermediate state of execution, even if the possible contribution of a given vertex $i.k$ into a maximum clique of G is completely unknown if $i.k$ lies in the unevaluated region, the contribution of $i.k$ in a maximum clique in $G^{i.k}$ is tightly estimated by $C[i][k]$ if the vertex lies in the evaluated region. Here, we propose an upper-bound on $|MCC(G)|$ that takes advantages of this property.

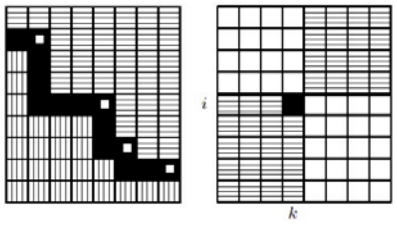


Fig. 2. Left: An intermediate state of DAST computation; **Right:** $G_L^{i.k}$

We define $G_L^{i.k}$, hereafter referred to as *local induced subgraph*, as the subgraph of G induced by the vertex set $V_L^{i.k} = V^{i+1.k+1} \cup \hat{V}^{i.k}$. Figure 2:Right, describes $G_L^{i.k}$ where $i.k$ is the black square.

Lemma 2. For any unevaluated cell $T[i][k]$ and any evaluated cell $T[j][l]$ such that $i < j$ and $k < l$, there exists a boundary cell $T[p][q]$ such that $i \leq p < j$ and $k \leq q < l$.

Proof. Consider the rectangle $R \subset T$ induced by cells $T[i][k]$ and $T[j][l]$ ($R = \{T[a][b]$ such that $i \leq a \leq j$ and $k \leq b \leq l\}$). By definition, R contains both unevaluated (at least $T[i][k]$) and evaluated cells (at least $T[j][l]$), so there exists an unevaluated cell $T[p][q] \in R$, which is adjacent to an evaluated cell, and since $T[p][q] \in R$, then $i \leq p < j$ and $k \leq q < l$.

Lemma 3. $MCC(G) = \max_{i.k|T[i][k] \text{ is boundary}} MCC(G_L^{i.k})$, and thus

$$\overline{MCC}(G) = \max_{i.k|T[i][k] \text{ is boundary}} \overline{MCC}(G_L^{i.k})$$

Proof. Proving Lemma 3 is equivalent to proving that the maximum clique lies in one of the *local induced subgraphs* of G that is induced by a *boundary cell*. Toward this goal, we will assume that we are in an intermediate state of execution, which implies that $T[1][1]$ is an unevaluated cell and that $T[m][n]$ has been evaluated (where m = number of rows in G, n = number of columns in G).

Any clique K in an alignment graph G is an increasing subset of vertices, namely, $K = \{i_1.k_1, i_2.k_2, \ldots, i_{|K|}.k_{|K|}\}$, where $i_l < i_{l+1}$ and $k_l < k_{l+1}$ for all $1 \leq l < |K|$. To prove that K lies completely inside one locally induced subgraph, we instead prove that $K' = \{i_0.k_0, i_1.k_1, \ldots, i_{|K|}.k_{|K|}, i_{|K|+1}.k_{|K|+1}\}$ lies completely inside one locally induced subgraph, where $i_0.k_0 = 1.1$ and $i_{|K|+1}.k_{|K|+1} = m.n$. Since K' intersects with both the *evaluated* and the *unevaluated* region, there exists l, such that vertices $i_0.k_0, i_1.k_1, \ldots, i_l.k_l$ lie in the unevaluated region and vertices $i_{l+1}.k_{l+1}, \ldots, i_{|K|}.k_{|K|}, i_{|K|+1}.k_{|K|+1}$ lie in the evaluated region. By invoking Lemma 2 with $i = i_l, k = k_l, j = i_{l+1}, l = k_{l+1}$, we obtain that there exists a boundary cell $T[p][q]$ such that $i_l \leq p < i_{l+1}$ and $k_l \leq q < k_{l+1}$. Thus, K' and hence the clique K lies entirely in the *local induced subgraph* induced by the boundary cell $T[p][q]$.

Since any clique in $G_L^{i.k}$ implicitly defines a clique over $\hat{V}^{i.k}$ (that is in the unevaluated region) and another clique over $V^{i+1.k+1}$ (that is in the evaluated region), $MCC(G_L^{i.k}) \leq MCC(\hat{G}^{i.k}) + MCC(G^{i+1.k+1})$. Then, $|MCC(G)|$ is upper-bounded by:

$$\overline{MCC}(G) = \max_{i.k|T[i][k] \text{ is boundary}} \overline{MCC}(\hat{G}^{i.k}) + \overline{MCC}(G^{i+1.k+1}), \tag{2}$$

where $\overline{MCC}(G^{i+1.k+1})$ is tightly estimated by $C[i+1][k+1]$, and where $\overline{MCC}(\hat{G}^{i.k})$ is estimated in a preprocessing step by using the longest increasing path in $\hat{G}^{i.k}$ (i.e. $\overline{MCC}(\hat{G}^{i.k}) = |LIP(\hat{G}^{i.k})|$).

Computing all $|LIP(\hat{G}^{i.k})|$) can be done in $O(n^2 \times m^2)$ time using the algorithm presented in [8]. Moreover, there are no more than $n+m$ boundary cells in T. The global upper-bound $\overline{MCC}(G)$ can be either computed once when the time limit is attained, or also can be maintained at each execution step for the need of branch and bounds strategy.

4 Computational Results

All presented results were obtained on a cluster under Linux RedHat Enterprise 5 architecture 64 Bits, 64 GB RAM, 2.8GHZ Intel Xeon. The efficiency of the dominance strategy for solving FIP was evaluated through two benchmarks. We tackled the FIP problem using the following protocol: any of the proteins has been considered as a query, then removed from the dataset and compared with the remaining proteins in order to find its family based on its nearest neighbor.

First, we used Skolnick set, described in [11]. It is a popular benchmark that contains 40 protein domains having from 97 to 256 residues and classified in SCOP (v1.73) into five families. The second benchmark comes from 3D SHape Recognition Contest 2010 (SHREC'10) [16] and consists of 50 query protein structures and 1000 target protein structures, all classified into 100 super-families in the CATH classification. The goal of this contest was to identify the family of each query. Identifying the 50 queries implies solving 50000 comparison instances. The best results have been obtained by the structure comparison tool A_purva [12]. We will us it to compare with the results of DAST.

4.1 DAST on Skolnick Set

Running time comparison. Computing the upper-bound at each intermediate state slowdowns the solving process. DAST$_a$ (without bounds computation) solves more instances than DAST$_b$ (with upper-bounds computation) when both methods are given the same distance threshold and the same time limit. For example, for a threshold of 3 Å and when the running time was bounded by 2 seconds per instance DAST$_a$ solved 506 instances, versus 338 for DAST$_b$ (i.e. DAST$_a$ is about 1.5 times faster than DAST$_b$). However, as we will see below, the advantages of DAST$_b$ for solving FIP using the dominances recompense notably this slowdown.

Solving FIP without dominance. Classifying all proteins from the Skolnick set without dominances (i.e. using DAST$_a$) requires solving 1560 instances. As shown in Table 1 when the running time was bounded by 2 seconds per instance, 1054 instances remained unsolved and none of the query could be assigned. Table 1 presents the evolution of the number of solved instances by DAST$_a$ with different time limits. Even with the larger time limit that we used (one hour per instance), 21 instances remained unsolved. The whole computation time was about 15 days, and all of the 29 queries that could be classified were correctly classified.

Solving FIP with dominance. As mentioned above, when the execution time was bounded by two seconds per instance, DAST$_b$ solved only 338 over 1560 instances.

Table 1. Three steps of the FIP computation over the Skolnick set. We use the following abbreviations: # ins –number of instances proceeded for the given lapse of time T (in seconds); # sol_ins –number of solved instances; # uns_ins –number of unsolved (unclassified) instances; # dom_ins –number of dominated instances when using DAST$_b$ and dominance; # ins_left –number of instances to reload; # ass_q –number of assigned (classified) queries.

		$DAST_a$			$DAST_b$			
Step	T (s)	# ins	# sol_ins	# uns_ins	# ins	# dom_ins	# ins_left	# ass_q
1	2	1560	506	1054	1560	1383	137	29 / 40
2	300	1054	767	287	137	122	15	37 / 40
3	3600	287	266	21	15	15	0	40 / 40

However, by applying the dominance relation, 29 queries where correctly assigned into their family by the nearest neighbor measure. Only 137 instances required further processing in order to complete the analysis. These instances were then reloaded with a larger time limit and this process was repeated until the family identification was fully completed. Table 1 details the 3 steps that were needed for Skolnick set. The entire computation time was about 5 hours and 45 minutes. So we observe that DAST$_b$ is significantly faster than DAST$_a$ when solving FIP. Moreover, using the dominance relations guarantees that the exact nearest neighbor is found **without solving all instances**, which is not true for DAST$_a$, neither for any algorithm that does not provide bounds.

4.2 DAST$_b$ versus A_purva on SHREC'10 Set

A_purva is an exact solver based on Contact Map Overlap maximization (CMO) similarity measure [12,17][1]. It has been shown to be both efficient (notably faster than the previous exact algorithms), and reliable (providing accurate upper and lower bounds of the solution). A_purva is based on an integer programming formulation of CMO, and it converges to the optimal solution using a branch and bound strategy. At each node, A_purva provides two numbers derived from a Lagrangian relaxation: a lowerbound LB and an upperbound UB of the maximum number of common contacts (ncc). When an instance is optimally solved, the relation $LB = UB$ holds. Otherwise, $UB > LB$ and the so called relative gap value $RG = (UB - LB)/UB$ gives the precision of the results. This property is very useful in the context of large-scale database comparisons where the execution time is usually bounded. The above properties make A_purva applicable for large-scale protein comparison and classification. Since it is an exact solver, it is often used to evaluate the quality of various heuristic approaches [18,19].

In this section, we compare DAST$_b$ with A_purva when solving FIP on SHREC'10 set. Both tools provide the best local matching (alignment) between proteins p_1 and p_2 in their corresponding feasible sets (compatible matching pairs), and according to their specific objective functions–maximum number of isometric pairs of amino-acids for DAST and, respectively, maximum number of common contacts for A_purva. On SHREC'10 dataset DAST$_b$ was more precise than A_purva. This can be explained by the isometric constraint in definition 1. Table 2 presents the different steps of this pro-

[1] A_purva is available at http://apurva.genouest.org

cess. A_purva assigned all the 50 queries, 46 of them were correctly predicted according to the CATH classification. However, A_purva failed for the queries 1tteA02, 1wwjA00, 1jftA01 and 3bioA02. DAST_b also assigned the 50 queries, and correctly predicted 49 (it failed for 3bioA02). However, A_purva was significantly faster than DAST_b (the corresponding total running time on SHREC'10 benchmark dataset was 28 hours versus 60 days).

Neither of the methods correctly classified the query 3bioA02. A_purva and DAST found two nearest neighbors from different families with similarity scores of 0.6059 and 0.2 respectively. These low values of DAST similarity score indicate that there is no true nearest neighbor for it in SHREC'10 data set. We contacted an expert from the domain[2] who confirmed the CATH classification of 3bioA02 and suggested us to study its similarity with protein 1f06. As a consequence, we observed that adding the domain 1f06A02 to SHREC'10 dataset allows to assign the query 3bioA02 correctly (i.e. SHREC'10 data set is not enough representative).

Figure 3 visualizes the alignments provided by A_purva and DAST for the query 1tteA02 which was wrongly predicted by A_purva but was correctly identified by DAST with 1ixrB03 as its nearest neighbor. Aiming to maximize the number of common contacts, A_purva matched non-isometric residues in the middle loop and at both ends. On the contrary, DAST matched closed sub-structures only (here the three helices of 1tteA02) and ignored the middle loop and extremities. For comparison purpose we also present here the alignment given by TM-align–well know protein structure comparator [20]. The TM-align alignment is very similar the DAST alignment.

Table 2. Number of dominated instances, of instances to reload and of assigned queries at each of the three steps of the FIP computation over the SHREC'10 set when using dominance, for both DAST_b and A_purva

Method	Step	Time limit (s)	# Instances	# Dominated	# Left	# Assigned queries
DAST_b	1	2	50000	41399	8551	12/ 50
	2	300	8551	7894	619	19/ 50
	3	3600	619	548	40	45 / 50
	4	7200	40	12	28	46 / 50
	5	60000	28	28	0	50 / 50
A_purva	1	2	50000	49721	229	43/ 50
	2	10	229	227	2	48/ 50
	3	50	2	2	0	50 / 50

Towards a combined tool. These results led us to propose a combined strategy for protein family identification. It uses the normalized *RMSDc*, defined as $NRMSDc = \frac{RMSDc}{\text{length(Query)}}$. First we ran A_purva on SHREC'10 set and computed the corresponding *NRMSDc* values. We observed that they were higher than 0.1 only for four instances (query,NN)–an indication for a strong deviation between the corresponding structures. For all other instances the *NRMSDc* value was obviously smaller, less than 0.05. We

[2] Alexey Murzin from the Laboratory of Molecular Biology, Cambridge.

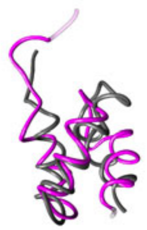
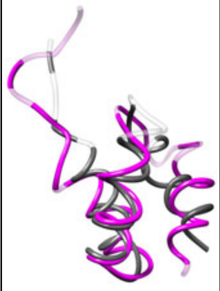
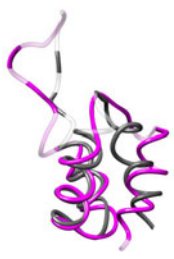
	A_purva	DAST	TM-align
			
len_align	53	43	45
cRMSD	5.35	2.37	2.71
TM-score	0.43	0.51	0.53

Fig. 3. The instance 1ixrB03-1tteA02 aligned by A_purva (left), DAST (center) and TM-align (right). The parameters for DAST were $\tau = 5.0$Å and sse1 (filter 1). The length of the associated alignment, (len_align), as well as the corresponding RMSDc and TM-score are given. We observe that A_purva matches as much as possible residues, while DAST and TM-align focuse on the local similar structures only.

also realized that these four instances correspond to the four wrongly predicted by A_purva couples (query-NN). Then DAST$_b$ was executed for these four queries only (it required computing new 4000 instances). This combined strategy achieved an accuracy of 50/50 correctly assigned queries (better than any of DAST or A_purva results) but for much less computational time than DAST running time.

5 Conclusion and Future Work

In this paper we enrich the local structure comparator tool DAST with bounds. This permits to use it in the context of a new dominance relation. The last one is very useful for the protein family identification problem since avoids solving all instances. Moreover, this relation is applicable to any NP-complete comparison methods and can be used for solving the FIP or for clustering large sets of protein structures.

Acknowledgements. Thanks to Inken Wohlers from CWI for discussions and suggestions. R. Andonov is supported by BioWIC ANR-08-SEGI-005 project. All computations were done on the Ouest-genopole bioinformatics platform (http://genouest.org)

References

1. Orengo, C., Thornton, J.: Protein families and their evolution - a structural perspective. Annual Review of Biochemistry 74(1), 867–900 (2005)
2. Koehl, P.: Protein structure similarities. Curr. Opin. Struct. Biol. 11(3), 348–353 (2001)
3. Gibrat, J.F., Madej, T., Bryant, S.: Surprising similarities in structure comparison. Current Opinion in Structural Biology 6, 377–385 (1996)

4. Gerstein, M., Levitt, M.: Using iterative dynamic programming to obtain accurate pair-wise and multiple alignments of protein structures. In: Proceedings of ISMB 1996, pp. 59–67 (1996)
5. Kabsch, W.: A discussion of the solution for the best rotation to relate two sets of vectors. Acta Crystallographica Section A 34(5), 827–828 (1978)
6. Holm, L., Sander, C.: Protein structure comparison by alignment of distance matrices. Journal of Molecular Biology 223, 123–138 (1993)
7. Godzik, A., Skolnick, J.: Flexible algorithm for direct multiple alignment of protein structures and seequences. CABIOS 10, 587–596 (1994)
8. Malod-Dognin, N., Andonov, R., Yanev, N.: Maximum cliques in protein structure comparison. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 106–117. Springer, Heidelberg (2010)
9. Lathrop, R.: The protein threading problem with sequence amino acid interaction preferences is np-complete. Protein Engineering 7(9), 1059–1068 (1994)
10. Karp, R.: Reducibility among combinatorial problems. Complexity of Computer Computations 6, 85–103 (1972)
11. Caprara, A., Carr, R., Israil, S., Lancia, G., Walenz, B.: 1001 optimal pdb structure alignments: integer programming methods for finding the maximum contact map overlap. J. Comput. Biol. 11(1), 27–52 (2004)
12. Andonov, R., Malod-Dognin, N., Yanev, N.: Maximum contact map overlap revisited. J. Comput. Biol. 18(1), 27–41 (2011)
13. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., Bourne, P.: The protein data bank. Nucleic Acids Research 28, 235–242 (2000)
14. Orengo, C., Michie, A., Jones, S., Jones, D., Swindells, M., Thornton, J.: Cath - a hierarchic classification of protein domain structures. Structure 5(8), 1093–1109 (1997)
15. Andreeva, A., Howorth, D., Chandonia, J.M., Brenner, S., Hubbard, T., Chothia, C., Murzin, A.: Data growth and its impact on the scop database: new developments. Nucl. Acids Res. 36, 419–425 (2007)
16. Mavridis, L., Venkatraman, V., Ritchie, D.W., Morikawa, N., Andonov, R., Cornu, A., Malod-Dognin, N., Nicolas, J., Temerinac-Ott, M., Reisert, M., Axenopoulos, H.B.A.: Shrec-10 track: Protein models. In: 3DOR: Eurographics Workshop on 3D Object Retrieval, pp. 117–124 (2010)
17. Malod-Dognin, N., Yanev, N., Andonov, R.: Comparing protein 3d structures using a_purva. Rapport de recherche RR-7464, INRIA (2010)
18. Di Lena, P., Fariselli, P., Margara, L., Vassura, M., Casadio, R.: Fast overlapping of protein contact maps by alignment of eigenvectors. Bioinformatics 26(18), 2250–2258 (2010)
19. Shibberu, Y., Holder, A.: A spectral approach to protein structure alignment. IEEE/ACM Trans. Comput. Biol. Bioinform. 8(4), 867–875 (2011)
20. Zhang, Y., Skolnick, J.: Tm-align: a protein structure alignment algorithm based on the tm-score. Nucleic Acids Research 33, 2302–2309 (2005)

Maximum Likelihood Estimation of Incomplete Genomic Spectrum from HTS Data

Serghei Mangul[1], Irina Astrovskaya[1], Marius Nicolae[2], Bassam Tork[1], Ion Mandoiu[2], and Alex Zelikovsky[1]

[1] Department of Computer Science, Georgia State University, Atlanta, GA 30303
[2] Department of Computer Science & Engineering, University of Connecticut, Storrs, CT 06269
{serghei,iraa,btork,alexz}@cs.gsu.edu,
{man09004,ion}@engr.uconn.edu

Abstract. High-throughput sequencing makes possible to process samples containing multiple genomic sequences and then estimate their frequencies or even assemble them. The maximum likelihood estimation of frequencies of the sequences based on observed reads can be efficiently performed using expectation-maximization (EM) method assuming that we know sequences present in the sample. Frequently, such knowledge is incomplete, e.g., in RNA-seq not all isoforms are known and when sequencing viral quasispecies their sequences are unknown. We propose to enhance EM with a virtual string and incorporate it into frequency estimation tools for RNA-Seq and quasispecies sequencing. Our simulations show that EM enhanced with the virtual string estimates string frequencies more accurately than the original methods and that it can find the reads from missing quasispecies thus enabling their reconstruction.

Keywords: high-throughput sequencing, expectation maximization, viral quasispecies, RNA-Sequencing.

1 Introduction

With the advent of high-throughput sequencing (HTS) technologies, it becomes possible to sequence samples containing multiple genomic sequences and then attempt to estimate their frequencies or even assemble them. In this paper we will consider two such HTS applications:

(i) RNA-seq, when the transcriptome (library of isoforms) is known but may be incomplete and expression of isoforms (or genes) is estimated by their frequencies in the sample and
(ii) viral quasispecies sequencing, when the reference sequence of the viral strain is known but the task is to find sequences of distinct quasispecies which are slightly different from the reference as well as to estimate their frequencies in the sample.

The maximum likelihood estimation of frequencies of the sequences (further referred as strings) can be efficiently performed using expectation-maximization (EM) method for the viral quasispecies application(see [4,10,1]) and for RNA-seq[1](see [7,8]). In brief, the input to EM consists of a panel, i.e., a bipartite graph in which one part correspond to the strings and another correspond to the reads. An edge connecting a read with a string expresses the possibility of the read to be emitted by the string with the probability associated with the edge. Given a panel and frequencies of the reads, EM can find maximum likelihood estimate of string frequencies.

Although in the both applications a certain knowledge about the sequences in the sample is available, such knowledge (recorded in the panel) is frequently incomplete. In case of RNA-seq, not all isoforms are already in the databases and in case of viruses, initially, no quasispecies sequences are known. In this paper, we propose a new method of enhancing EM that tries to estimate the incompleteness of the panel obtaining more accurate estimates of string frequencies and identifying reads that are more probable to be emitted by missed strings.

The method adds a *virtual string* to the panel and then iteratively changes the panel by assigning reads to the virtual string. The proposed enhanced method, so called Virtual String EM (VSEM), has been incorporated into IsoEM [8] and ViSpA [1]. Our simulations show that the VSEM-enhanced methods (IsoVSEM and ViSpA-VSEM) estimate string frequency more accurately than the original methods and that ViSpA-VSEM can find the reads from missing quasispecies thus enabling their reconstruction.

The rest of the paper is organized as follows. The next section describes VSEM. In Section 3 we describe the IsoVSEM and results of its experimental validation on transcriptome libraries. Section 4 describes the combination ViSpA-VSEM of ViSpA and VSEM. In Section 5, we analyze experimental results comparing ViSpA, ViSpA-VSEM and ShorAH [10] on the simulated reads with and without sequencing errors.

2 Virtual String Expectation Maximization

In this section we first formally define the panel and briefly describe EM method. Then we show how to estimate the quality of the model. Finally we describe the VSEM method enhancing EM with the virtual string.

The input data for EM method consists of a *panel*, i.e., a bipartite graph $G = \{S \bigcup R, E\}$ such that each string is represented as a vertex $s \in S$, and each read is represented as a vertex $r \in R$. With each vertex $s \in S$, we associate unknown frequency f_s of the string. And with each vertex $r \in R$, we associate observed read frequency o_r. Then for each pair s_i, r_j, we add an edge (s_i, r_j) weighted by probability of string s_i to emit read r_j with m genotyping errors:

$$h_{s_i,r_j} = \binom{l}{m} (1-\epsilon)^{l-m} \epsilon^m,$$

where l is length of read sequence, and ϵ is the genotyping error rate.

Regardless of initial conditions EM algorithm always converge to maximum likelihood solution (see [3]).The algorithm starts with the set of N strings. After uniform initialization of frequencies $f_s, s \in S$, the algorithm repeatedly performs the next two steps until convergence:

[1] Note that frequency estimation based on previous approaches is less accurate (see e.g. [9]).

- E-step: Compute the expected number $n(j)$ of reads that come from string i under the assumption that string frequencies $f(j)$ are correct, based on weights $h_{s_i,j}$
- M-step: For each i, set the new value of f_s to to the portion of reads being originated by string s among all observed reads in the sample

In order to decide if the panel is incomplete we need to measure how well maximum likelihood model explains the reads. We suggest to measure the model quality by the deviation between expected and observed read frequencies as follows:

$$D = \frac{\sum_j |o_j - e_j|}{|R|},$$

where $|R|$ is number of reads, o_j is the observed read frequency of the read r_j and e_j is the expected read frequencies of the read r_j calculated as follows:

$$e_j = \sum_i \frac{h_{s_i,j}}{\sum_l h_{s_i,l}} f_i^{ML} \qquad (1)$$

where $h_{s_i,j}$ is weighted match based on mapping of read r_j to string s_i and f_j^{ML} is the maximum-likelihood frequency of the string s_i.

The main idea of the VSEM algorithm (see Algorithm 1) is to add to set of candidate strings a virtual string which emits reads that do not fit well to existing sequences. The flowchart of VSEM is on Fig. 1. Initially, all reads are connected to the virtual string with weight $h_{s_i,j} = 0$. The first iteration finds the ML frequency estimations of candidates strings, ML frequency estimations of virtual string will be equal to 0, since all edges between virtual string and reads $h_{vs,j} = 0$. Then these estimation are used to compute expected frequency of the reads according to (1). If the expected read

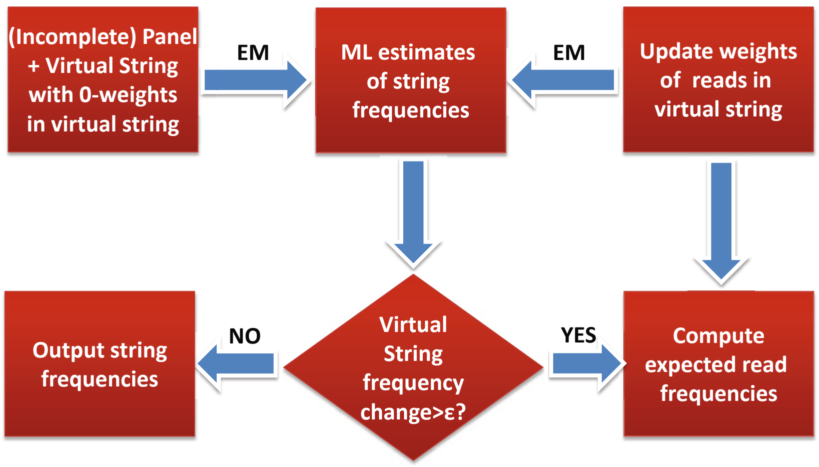

(Incomplete) Panel + Virtual String with 0-weights in virtual string
EM
ML estimates of string frequencies
EM
Update weights of reads in virtual string
Output string frequencies
NO
Virtual String frequency change>ε?
YES
Compute expected read frequencies

Fig. 1. Flowchart for VSEM

Algorithm 1. VSEM algorithm

```
add virtual string $vs$ to the set of candidate strings
initialize weights $h_{vs,j} = 0$
while $\Delta vs > \epsilon$ do
    calculate $f_j^{ML}$ by EM algorithm
    $e_j = \sum_i \frac{h_{s_i,j}}{\sum_l h_{s_i,l}} f_i^{ML}$
    $D = \frac{\sum_j |o_j - e_j|}{|R|}$
    $\delta = o_j - e_j$
    if $\delta > 0$ then
        $h_{vs,j} += \delta$
    else
        $h_{vs,j} = \max\{0, h_{vs,j} + \delta\}$
    end if
end while
```

frequency is less than the observed one (under-estimated), then the lack of the read expression is added to the weight of the read connection to the virtual string. For over-estimated reads, the excess of read expression is subtracted from the corresponding weight (but keeping it non-negative). The iterations are continued while the virtual string frequency is decreasing by more than ϵ.

Enhancing of ViSpA. Resulted edge weight between virtual string and each read can be interpreted as the probability of the read to be emitted by missing strings. VSEM transmits to ViSpA assembler the rounded read weights and during assembling of additional candidate quasispecies the preference is given to reads which are likely emitted by missing strings.

Enhancing of IsoEM. The IsoEM incorporates the virtual string and the resulting isoform frequency estimations are improved. Based on the frequency of virtual string it is possible to decide if the panel is likely to be incomplete, the total frequency of missing strings is estimated by frequency of virtual string.

3 Experimental Validation of IsoVSEM on RNA-Seq Data

IsoEM is a novel expectation-maximization algorithm for inference of alternative splicing isoform frequencies from high-throughput transcriptome sequencing (RNA-Seq) data proposed in [8]. IsoEM takes advantage of base quality scores, strand information and exploits disambiguation information provided by the distribution of insert sizes generated during sequencing library preparation. In the bipartite graph consisting of isoforms and reads an edge from an isoform to a read represents possibility that a read is emitted by the isoform. It is noted [8] that EM can run in parallel for each connected component of this bipartite graph. We enhance IsoEM algorithm by adding virtual string to each connected component. The resulted algorithm IsoVSEM in the nested loop applies IsoEM instead of EM (see Algorithm 1). Since isoforms have different length we estimate missing isoforms by volume defined as frequency of isoform multiplied by its length.

Our validation of IsoVSEM includes two experiments over human RNA-seq data. Below we describe the transcriptome data and read simulation and then give the settings for the each experiment and analyze the obtained experimental results.

Data sets. IsoVSEM was tested on human RNA-Seq data. The human genome data(hg18, NCBI build 36) was downloaded from UCSC and CCDS together with the coordinates of the isoforms in the KnownGenes table. The UCSC database contains 66803 isoforms from 19372 genes, and CCDS database contains 20829 isoforms from 17373 genes. Genes were defined as clusters of known isoforms defined by the GNFAtlas2 table such that CCDS data set can be identified with the subset of UCSC data set.

30M single error-free reads of length 25 were randomly generated by sampling fragments of isoforms from UCSC data set. Each isoform was assigned a true frequency based on the abundance reported for the corresponding gene in the first human tissue of the GNFAtlas2 table, and a probability distribution over the isoforms inside a gene cluster [8]. We simulate datasets with geometric (p=0.5) distributions for the isoforms in each gene.

Table 1. Median percent error (MPE) and 15% error fraction ($EF_{.15}$) for isoform expression levels in Experiment 1

	Expression range	0	$(0, 10^{-6}]$	$(10^{-6}, 10^{-5}]$	$(10^{-5}, 10^{-4}]$	$(10^{-4}, 10^{-3}]$	$(10^{-3}, 10^{-2}]$	All
	Full panel	0.0	61.7	22.0	8.0	3.2	2.1	10.3
MPE	Incomplete	0.0	59.3	41.3	24.8	19.7	5.9	33.7
	Incomplete + VS	0.0	47.2	33.1	20.7	16.4	8.5	26.9
$EF_{.15}$	Full panel	0.0	81.9	61.3	28.7	7.5	8.5	38.8
	Incomplete	0.0	81.7	72.4	61.4	56.7	42.1	67.6
	Incomplete+VS	0.0	77.2	68.2	57.6	53.0	36.8	63.6

Experiment 1: Comparison between IsoEm and IsoVSEM on reduced transcriptome data. We assumed that in every gene 25% of isoforms is missing. In order to create such an instance we assign to isoforms inside the gene geometric distribution(p=0.5), assuming a priori that number of isoforms inside the gene is less or equal to 3. This way we removed isoform with frequency 0.25. As a result 11339 genes were filtered out, number of isoforms was reduced to 24099. Note that in our data set missing isoforms do not have unique exon junctions that can emit reads indicating that certain isoforms are missing.

We first check how well IsoVSEM can estimate the volume of missing strings. Although the frequencies of all missing strings (isoforms) is the same (25%), the volumes significantly differ because they have different length. Therefore, the quality can be measured by correlation between actual missing volumes and predicted missing volumes which are volumes of virtual strings. In this experiment it is 61% which is sufficiently high to give an idea which genes are missing isoforms in the database.

Table 1 reports the median percent error (MPE) and .15 error fraction $EF_{.15}$ for isoform expression levels inferred from 30M reads of length 25, computed over groups

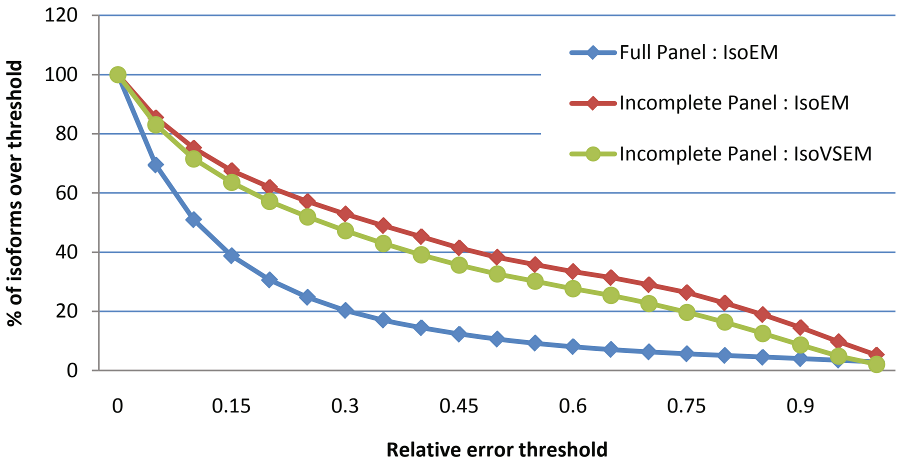


Fig. 2. Error fraction at different thresholds for isoform expression levels inferred from 30M reads of length 25 simulated assuming geometric isoform expression. Blue line correspond to IsoEM with the full panel, read line is IsoEM with the incomplete panel, and green line is IsoVSEM.

of isoforms with various expression levels. MPE is the median relative error of isoform frequency estimation and the error fraction with threshold t, denoted EF$_t$, is defined as the percentage of isoforms with relative error greater or equal to t.

Figure 2 gives the error fraction at different thresholds ranging between 0 and 1. Clearly the best performance is achieved when the the isoform library is full, using virtual string explains accuracy gain of IsoVSEM over IsoEM. IsoVSEM achieves better accuracy in the case when the panel is incomplete. Performance of IsoEm and IsoVSEM for the full panel is the same.

Experiment 2: Comparison between IsoEm and IsoVSEM on the CCDS panel. In this experiment UCSC database represents the full set of isoforms and CCDS represents the incomplete panel. Reads were generated from UCSC library of isoforms, while only frequencies of known isoforms from CCDS database were estimated. In contrast to

Table 2. Median percent error (MPE) and 15% error fraction (EF$_{.15}$) for isoform expression levels in Experiment 2

Expression range		0	$(0, 10^{-6}]$	$(10^{-6}, 10^{-5}]$	$(10^{-5}, 10^{-4}]$	$(10^{-4}, 10^{-3}]$	$(10^{-3}, 10^{-2}]$	All
	Full panel	0.0	100	22.7	7.3	3.5	2.5	11.8
MPE	Incomplete	0.0	100	45.5	29.4	28.5	28.7	31.8
	Incomplete + VS	0.0	100	43.2	27.09	25.68	14.34	29.61
EF$_{.15}$	Full panel	5.1	91.2	62.8	29.3	15.8	7.6	45.5
	Incomplete	18.6	95.6	85.6	83.3	89.2	86.7	80.0
	Incomplete+VS	17.6	91.8	81.3	77.9	80.3	75.5	75.2

Experiment 1, we do not control the frequency of missing isoforms (i.e., isoforms from UCSC which are absent in CCDS). Therefore, one cannot expect as good improvements as in Experiment 1.

Table 2 reports the median percent error (MPE) and .15 error fraction $EF_{.15}$ for isoform expression levels inferred from 30M reads of length 25, computed over groups of isoforms with various expression levels. We do not report the number of isoforms since they are different for UCSC and CCDS panels. Anyway, one can see a reasonable improvement in frequency estimation of IsoVSEM over IsoEM.

4 VSEM Enhancement of ViSpA

In this section we first give high-level description of ViSpA [1], a recent viral spectrum assembling tool for inferring viral quasispecies sequences and their frequencies from pyrosequencing shotgun reads. Then we describe the flowchart of the combining tool ViSpA-VSEM and required modifications to ViSpA.

ViSpA: Viral Spectrum Assembly. First, ViSpA aligns the reads to the consensus genome sequence using SEGEMEHL [6] software correcting obvious sequencing errors and removes subreads (reads that are completely covered by larger reads). Then it builds a read graph with vertices representing remaining reads (superreads) and edges representing overlaps between them. In this graph, each path from the leftmost vertex to the rightmost vertex corresponds to a possible candidate quasispecies sequence. For each edge e, ViSpA computes probability $p(e)$ to connect two reads from the same quasispecies. Then ViSpA assigns cost $-\log(p(e)) = \log(1/p(e))$ to each edge e, making the minimum-cost paths more probable to represent quasispecies sequences. Next, a set of candidate paths consisting of the max-bandwidth paths (paths minimizing maximum edge cost) through each vertex is created and refined so that only distinct sequences remain. The maximum-likelihood estimates of frequencies are calculated by EM algorithm which takes in account all reads in the sample. Finally, ViSpA reports most frequent candidate sequences and their frequencies as inferred viral quasispecies spectrum.

Combining ViSpA with VSEM. Knowing which reads in a sample are likely to be produced from missing (unknown) quasispecies sequences may allow to expand ViSpA's candidate set. Additionally, we can improve ViSpA's estimates for quasispecies frequencies by taking in account incompleteness of the panel.

Figure 3 illustrates the proposed workflow between (modified) ViSpA and VSEM. At each iteration, ViSpA gets a set of aligned reads and their 0/1-weights estimating probability to be emitted by unknown strings (candidates). Initially, all reads have weight zero and ViSpA works as described above except the maximum likelihood estimates for candidate quasispecies sequences (strings) are calculated by VSEM. On all other iterations, VSEM panel includes not only newly assembled sequences but also candidate sequences stored from the previous iterations. All sequences with negligible EM frequency are filtered out from the cumulative set of sequences, called quasispecies library. Once frequencies distribution is obtained, VSEM assigns 0/1-weight for each read: 0 corresponds to high probability to be emitted by a missing sequences and 1,

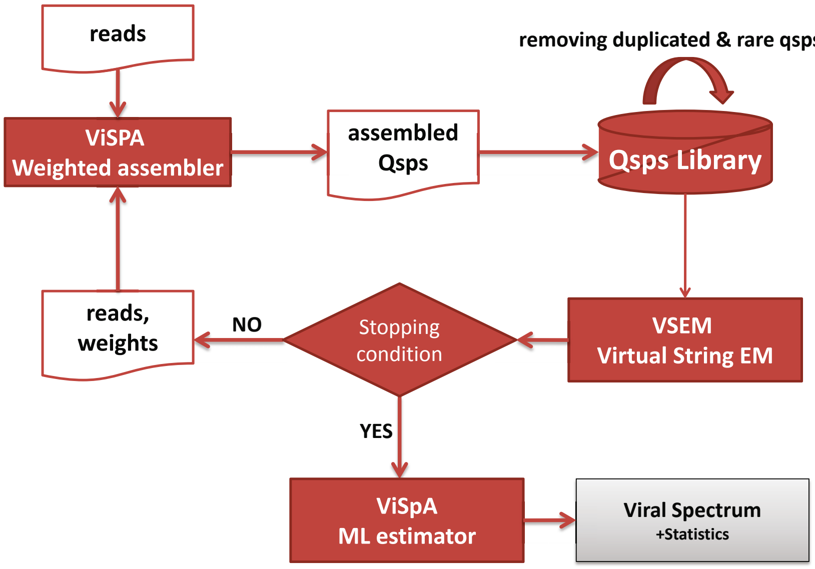


Fig. 3. Flowchart of ViSpA-VSEM

otherwise. Finally, if the virtual string has high EM frequency and we expand our quasispecies library with respect to previous iteration, we feed reads and their weights back to ViSpA, and all process is repeated. At the end, we report sequences from quasispecies library and their frequencies as reconstructed viral quasispecies spectrum.

The modifications of ViSpA in ViSpA-VSEM include (1) superread selection (a weight-1 superread is removed if it shares a subread with a 0-weight superread), (2) edge cost computation which takes in account vertex weights:

$$cost'(e) = cost(e) + 0.5 \cdot L \cdot (w(u) + w(v)),$$

where $cost(e)$ is the original cost of e, L is the read length, and $w(u)$, $w(v)$ are the 0/1 weights assigned by VSEM to read v.

5 Experimental Validation of ViSpA-VSEM on Simulated Data

Data Sets. We simulate reads from 1739-bp long fragment from the E1E2 region of 44 HCV sequences [5]. Each population was created by randomly selecting either 10 or 40 sequences among these HCV variants and assigning frequencies following either (1) uniform (all sequences have the same frequency), or (2) skewed uniform (a single sequence has high frequency; all other sequences have uniformly low frequency), or (3) geometric (the i^{th} sequence is a constant percentage more frequent that the $(i+1)^{th}$ sequence) distributions.

First, we simulate error-free reads without indels with respect to the reference sequence. The length of a read follows normal distribution with variance 400, and starting position follows the uniform distribution. This simplified model of reads generation has two parameters: number of the reads that varies from 20K up to 100K and the averaged read length that varies from 100bp up to 500bp.

Then we simulate 454 pyrosequencing reads from the 10 random quasispecies (under geometric distribution) out of 44 HCV sequences [5] using FlowSim [2]. The generated dataset contains 39,131 reads with length varying from 50bp up to 550bp and average length equaled to 322bp. Each position (except the end) is covered by at least 4000 reads. 99.96% of aligned reads has at least one indel with respect to the reference: 99.97% of deletions and 99.6% of insertions are 1bp long. Only 1.1% of aligned reads have unknown base(s).

Frequency estimation quality. We evaluate predicted frequencies by the following statistics.

- Kullback-Leibler divergence

$$RE = \sum_{i \in I} p_i \log \frac{p_i}{q_i},$$

  where $P = \{p_i\}$ and $Q = \{q_i\}$ are true distribution and its approximation, and $I = \{i | p_i > 0,\, q_i > 0\}$ are real sequences among assembled candidate sequences,
- correlation between real and predicted frequencies,
- average prediction error:

$$\overline{err} = \frac{\sum_{i \in I} |p_i - q_i|}{|I|}.$$

Detection of panel incompleteness. We have checked how well VSEM can detect incompleteness of the panel in the following experiment. We have repeatedly (for different simulated frequency distribution for 10 quasispecies strings) deleted from the full panel each string (one at a time) and record the resulted frequency of the virtual string. If no string has been deleted, then virtual string has always stopped growing at frequency less than 10^{-6}, and if the frequency of the deleted string has been at least 1%, then the resulted virtual string frequency has grown to at least .5%. Thus VSEM can reliably detect incomplete panel if missing strings have total frequency at least 1%.

Improving quasispecies frequencies estimation using VSEM. Fig. 3 show our experimental results on simulated error-free reads generated from 40 quasispecies. The correlation is slightly improved for cases when the portion of missing strings is small and increases to as much as 15% when bigger portion of strings is missing.

The results on reads generated by Flowsim [2] and corrected by ShorAH are very similar to the results on error free reads.

Detection of reads emitted by missing strings. The output of VSEM besides estimated frequency of the virtual string also contain the weights of edges connecting reads to the virtual string. These weights can be interpreted as probabilities of reads to be emitted by the missing strings. In our experiments we have repeatedly measure the correlation

Table 3. Correlation (r) and average prediction error (err) between real quasispecies frequencies and estimated quasispecies frequencies for EM vs VSEM. r.l/n.r denote the read length / number of reads

	r.l./n.r	% of missing strings											
		$< 10\%$		10%-20%		20%-30%		30%-40%		40%-50%		$> 50\%$	
		r	err	r	err	r	err	r	err	r	err	r	err
ViSpA	100/20K	90.2	4.5	91.0	6.8	75.4	5.1	68.6	1.6	40.8	2.3	39.8	10.4
ViSpA-VSEM	100/20K	91.6	2.3	92.8	4.4	76.5	4.1	70.5	1.4	54.2	2.0	50.8	7.4
ViSpA	300/20K	95.7	3.8	93.2	10.2	89.8	1.0	66.7	1.5	62.1	2.1	46.8	9.7
ViSpA-VSEM	300/20K	95.4	1.7	95.8	1.1	96.9	0.6	85.7	0.9	88.0	0.9	60.4	2.6
ViSpA	100/100K	95.2	4.5	93.9	9.1	84.8	1.4	74.2	1.8	74.5	2.3	73.4	9.9
ViSpA-VSEM	100/100K	97.8	2.6	95.6	3.0	86.3	1.3	79.8	1.7	79.0	2.1	74.2	8.8
ViSpA	300/100K	96.2	3.9	88.6	12.4	88.9	1.0	85.1	1.4	75.1	2.3	49.5	10.5
ViSpA-VSEM	300/100K	96.2	2.0	92.8	0.9	93.7	0.7	90.2	1.2	84.4	1.7	67.1	4.8

between the edge weights and the spectrum of reads emitted by missing strings which has always exceeded 65%.

ViSpA versus ViSpA-VSEM. We compare quality of assembling and frequency estimation for both methods. Quality of assembling is measured by sensitivity (portion of the assembled real sequences among all real quasispecies) and its positive predictive value (portion of the real sequences among all assembled) in cross-validation tests.

Error-free reads. Previously [1], we demonstrate that ViSpA outperforms SHORAH in assembling haplotypes on error-free reads. ViSpA-VSEM can further improve predictive power and frequency estimation of ViSpA (see Table 4). On average, it infers additional two (in case of geometric distribution) to four sequences (in case of uniform and skewed uniform distributions). Taking into account unknown quasispecies sequences allows ViSpA-VSEM to estimate frequencies more accurately (average error is decreased 2.5 times for geometric distribution and more than 5 times for skewed uniform and uniform distributions). Since relative entropy and correlation coefficient r are measured only on the correctly inferred quasispecies sequences and are not adjusted with respect to the number of all quasispecies sequences in a sample, increasing relative entropy and decreasing of correlation coefficient r are not correlated with loss of predictive power. For example, predictive power is improved by obtaining additional real quasispecies in the case of geometric distribution whereas correlation coefficient becomes smaller.

Reads with simulated genotyping errors. It has been shown that ViSpA outperforms SHORAH if sequencing errors are initially corrected (see [1]). So in our experiments, we compare ViSpA and ViSpA-VSEM only on ShoRAH-corrected reads (see Fig. 5). The table reports the difference between 10 most frequent assemblies obtained by ViSpA and 10 most frequent assemblies obtained after two iterations of ViSpA-VSEM. ViSpA-VSEM can additionally infer a real quasispecies without allowing any mismatches between sequences($k = 0$). Again, the frequency estimation is more accurate

Table 4. Comparison between ViSpA and ViSpA-VSEM. Experiments are run on 100K reads from 10 quasispecies with average read length equaled to 300. The table reports PPV, sensitivity(SE), relative entropy (RE), correlation between real and predicted frequencies (r), and averaged prediction error ($\overline{err}$)(reported in %). "Gain" column reports averaged number of additionally inferred real quasispecies sequences after 4 iterations (on average) for skewed distribution, 5 iterations (on average) for geometric distribution and 13 iterations (on average) for uniform distribution.

	ViSpA					ViSpA-VSEM					
Distribution	PPV	SE	RE	r	err	PPV	SE	RE	r	err	Gain
Geometric	0.767	0.5	-0.0099	0.954	7.36	0.5905	0.73	0.0276	0.9094	2.91	2.3
Skewed	0.733	0.4	-0.0196	0.6725	13.01	0.701	0.77	0.0085	0.9665	2.5	4
Uniform	0.733	0.4	-0.0191	0.716	12.76	0.645	0.73	0.0108	0.9762	2.34	3.7

Table 5. Comparison between ViSpA and ViSpA-VSEM on their 10 most frequent assemblies. Experimental results are run on 100K reads from 10 quasispecies with average read length equaled to 300. The quasispecies sequence is considered found if one of candidate sequences matches it exactly ($k = 0$) or with at most k (2, 6 or 7) mismatches. The table reports PPV, sensitivity(SE), relative entropy (RE), correlation between real and predicted frequencies (r), and averaged prediction error ($\overline{err}$)(reported in %). "Gain" column reports averaged number of additionally inferred real quasispecies sequences after 2 iterations.

	ViSpA					ViSpA-VSEM					
#mismatches	PPV	SE	RE	r	err	PPV	SE	RE	r	err	Gain
k = 0	0.5	0.5	0.0720	0.9860	9.98	0.5455	0.6	0.0494	0.9741	7.54	1
k = 2	0.6	0.6	0.0668	0.9860	9.16	0.6364	0.7	0.0434	0.9680	6.67	1
k = 6	0.7	0.7	0.0577	0.9856	7.95	0.7273	0.8	0.0369	0.9463	6.20	1
k =7	0.8	0.8	0.0525	0.9866	7.26	0.8182	0.9	0.0335	0.9479	5.65	1

since ViSpA-VSEM EM takes into account missing quasispecies which is confirmed by the drop of the average prediction error.

6 Conclusions and Future Works

In this paper, we propose VSEM, a novel modification of EM algorithm which allows to estimate the frequencies of multiple genomic sequences present in a sample sequenced with HTS technology. VSEM is aimed to improve the maximum likelihood frequency estimations of assembled sequences and identify reads that belong to unassembled sequences. We have applied VSEM to enhance two tools: IsoEM (for inferring isofrom expression from RNA-seq data) and ViSpA (for inferring viral quasispecies spectrum from pyrosequencing shotgun reads). Our experimental study shows that VSEM-enhanced tools significantly improve their performance: IsoVSEM has better accuracy in estimation isoform frequencies and ViSpA-VSEM can infer more quasispecies sequences and better estimate their frequencies. Our results show potential of VSEM to improve other metagenomics tools.

Acknowledgements. SM, IA and BT were supported in part by Molecular Basis of Disease Fellowship, Georgia State University. MN and IIM were supported in part by NSF awards IIS-0546457, IIS-0916948, and DBI-0543365. SM, IA, BT and AZ were supported in part by NSF award IIS-0916401. SM, IA, MN, BT, IM and AZ were supported in part by Agriculture and Food Research Initiative Competitive Grant no. 2011-67016-30331 from the USDA National Institute of Food and Agriculture.

References

1. Astrovskaya, I., Tork, B., Mangul, S., Westbrooks, K., Mandoiu, I., Balfe, P., Zelikovsky, A.: Inferring viral spectrum from 454 pyrosequencing reads. BMC Bioinformatics (to appear), http://dna.engr.uconn.edu/bibtexmngr/upload/Aal.11a.pdf
2. Balser, S., Malde, K., Lanzen, A., Sharma, A., Jonassen, I.: Characteristics of 454 pyrosequencing data–enabling realistic simulation with flowsim. Bioinformatics 26, i420–i425 (2010)
3. Zaitlen, N., Pasaniuc, B., Halperin, E.: Accurate estimation of expression levels of homologous genes in RNA-seq experiments. Journal of Computational Biology 18(3), 459–468 (2011)
4. Eriksson, N., Pachter, L., Mitsuya, Y., Rhee, S.Y., Wang, C.: et al. Viral population estimation using pyrosequencing. PLoS Comput. Biol. 4, e1000074 (2008)
5. Von Hahn, T., Yoon, J.C., Alter, H., Rice, C.M., Rehermann, B., Balfe, P., Mckeating, J.A.: Hepatitis c virus continuously escapes from neutralizing antibody and t-cell responses during chronic infection in vivo. Gastroenterology 132, 667–678 (2007)
6. Hoffmann, S., Otto, C., Kurtz, S., Sharma, C.M., Khaitovich, P., Vogel, J., Stadler, P.F., Hackermüller, J.: Fast mapping of short sequences with mismatches, insertions and deletions using index structures. PLoS Comput. Biol. 5(9), e1000502 (2009)
7. Li, B., Ruotti, V., Stewart, R.M., Thomson, J.A., Dewey, C.N.: RNA-Seq gene expression estimation with read mapping uncertainty. Bioinformatics 26(4), 493–500 (2010)
8. Nicolae, M., Mangul, S., Mandoiu, I.I., Zelikovsky, A.: Estimation of alternative splicing isoform frequencies from RNA-seq data. Algorithms for Molecular Biology 6, 9 (2011)
9. Mortazavi, A., Williams, B.A.A., McCue, K., Schaeffer, L., Wold, B.: Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nature methods (2008)
10. Zagordi, O., Geyrhofer, L., Roth, V., Beerenwinkel, N.: Deep sequencing of a genetically heterogeneous sample: local haplotype reconstruction and read error correction. Journal of Computational Biology: A Journal of Computational Molecular Cell Biology 17(3), 417–428 (2010)

Algorithm for Identification of Piecewise Smooth Hybrid Systems: Application to Eukaryotic Cell Cycle Regulation

Vincent Noel[1,2], Sergei Vakulenko[4], and Ovidiu Radulescu[2,3]

[1] Université de Rennes 1 - CNRS UMR 6625 (IRMAR), Campus de Beaulieu, 35042 Rennes, France
[2] Université de Montpellier 2, DIMNP - UMR 5235 CNRS/UM1/UM2, Pl. E. Bataillon, Bat 24, CP 107, 34095 Montpellier Cedex 5, France
[3] INRIA Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes, France
[4] Institute of Print, St. Petersburg, Russia

Abstract. We discuss piecewise smooth hybrid systems as models for regulatory networks in molecular biology. These systems involve both continuous and discrete variables. The discrete variables allow to switch on and off some of the molecular interactions in the model of the biological system. Piecewise smooth hybrid models are well adapted to approximate the dynamics of multiscale dissipative systems that occur in molecular biology. We show how to produce such models by a top down approach that use biological knowledge for a guided choice of important variables and interactions. Then we propose an algorithm for fitting parameters of the piecewise smooth models from data. We illustrate some of the possibilities of this approach by proposing hybrid versions of eukaryotic cell cycle regulation.

Keywords: systems biology, hybrid models, cell cycle.

1 Introduction

Hybrid systems are widely used in automatic control theory to cope with situations arising when a finite-state machine is coupled to mechanisms that can be modeled by differential equations [MS00]. It is the case of robots, plant controllers, computer disk drives, automated highway systems, flight control, etc. The general behavior of such systems is to pass from one type of smooth dynamics (mode) described by one set of differential equations to another smooth dynamics (mode) described by another set of differential equations. The command of the modes can be performed by one or several discrete variables. The mode change can be accompanied or not by jumps (discontinuities) of the trajectories.

Depending on how the discrete variables are changed there may be several types of hybrid systems: switched systems [SWM+07], multivalued differential automata [Tav87], differential equations with discontinuous vector fields [FA88], piecewise affine [DJGH+04] and piecewise smooth systems [DB08]. Notice that in the last two cases, the mode changes when the trajectories attain some smooth manifolds.

Piecewise affine hybrid systems have been used to model dynamics of gene networks [DJGH+04,GRW07]. In these models, the gene variables evolve towards discrete values (attractors). The transient dynamics leading to attractors is considered to be piecewise affine where the linear part of the dynamical equations is defined by a diagonal matrix with negative entries. The transitions between discrete attractors are dictated by the relative position of the above variables with respect to some thresholds. Piecewise-affine hybrid models can be effectively used in verification studies, for instance computing the set of reachable states of a model. Identification of piecewise-affine models is a difficult problem approached elsewhere with various methods such as affine approximations of vector fields [DMT10] and discrete/continuous optimization algorithms [BBG00].

Although sufficient for certain applications like gene networks, piecewise affine models are less adapted to describe phenomena where the dynamics between two successive discrete events is strongly nonlinear. A typical example of such phenomena is the machinery of the cell cycle. Proteolytic degradation of the cyclins is switched on rapidly by the cyclin dependent kinase complexes but between two successive switchings the complexes have non-linear dynamics implying several positive (autocatalytic processes) and negative feed-back loops. These non-linear processes contribute to the robustness of the mechanism.

The idea of piecewise smooth systems arises naturally in the context of biochemical systems with multiple separated timescales. The dynamics of a multiscale, dissipative, large model, can be reduced to the one of a simpler model, called dominant subsystem [RGZL08, GRZ10, GR08]. The dominant subsystem depends on the comparison among the time scales of the large model. For nonlinear models, the dominant subsystem (which can be assimilated to a mode) is only piecewise constant and can change several times during the dynamics. The model reduction methods proposed in [GR08, RGZL08] generate dominant subsystems whose reactions rates are multivariate monomials of the concentration variables, like in the well-known S-systems [SV87]. Indeed, when applied to models using mass action kinetics, quasi-steady state and quasi-equilibrium approximations [GRZ10] lead to lumped models in which the reactions rates result from solving systems of polynomial equations. In general, these polynomials contain only a few terms (fewnomials). The solutions of such systems are much simplified in the case of total separation of the nonconstant terms in the fewnomials and lead to monomial rates. The rate of the same reaction can be represented by different monomials in different dominant subsystems (modes). For instance, the rate of a Michaelis-Menten mechanism depends linearly on the concentration of the substrate for small concentrations and is constant at saturation. We expect that more general rate laws [LUK10] can be treated similarly in our approach.

In this paper we propose a heuristic to construct appropriate modes and adequate piecewise smooth models by using a top-down approach. Then, we show how the parameters of the hybrid model can be identified from data or from trajectories produced by existing smooth, but more complex models.

2 Hybrid Models

We consider the so-called hybrid dynamical systems (HDS) consisting of two components: a continuous part, u, satisfying the equations

$$\frac{du_i}{dt} = f_i(u(t), s(t)), \quad t > 0, \tag{2.1}$$

where $u(t) = (u_1(t), u_2(t), ..., u_n(t)) \in \mathbf{R}^n$, and a discrete part $s(t) \in S$, where S is a finite set of states. We consider that there is an increasing series $\tau_0 = 0 < \tau_1 < \ldots < \tau_k < \ldots$ such that the discrete variables are piecewise constant on the intervals $[\tau_i, \tau_{i+1}[$ and that they change values at $t = \tau_k$. The continuous variables can also have discrete jumps at $t = \tau_k$.

Typically, in molecular networks, the continuous variables are protein concentrations and the discrete states may be gene or protein activities described by boolean variables $s(t) = (s_1(t), s_2(t), ..., s_m(t))$, where $s_j(t) \in \{0, 1\}$.

There are several possible ways to define the evolution of the s variables. Rather generally, this can be done by a time continuous Markov chain with transition probabilities $p(s, s', u)$ from the state s to the state s' (per unit time) depending on current state $u(t)$. However, in many molecular regulatory networks, transition probabilities dependence on u is not smooth. For instance, the probability for s to jump is close to one if u goes above some threshold value, and close to zero if u is smaller than the threshold. We can, in certain cases, neglect the transition time with respect to the time needed for u variables to change. Assuming that some of the discrete variables contribute to production of u and that other contribute to the degradation of u we obtain a general model of hybrid piece-wise smooth dynamical system

$$\frac{du_i}{dt} = \sum_{k=1}^{N} s_k P_{ik}(u) + P_i^0(u) - \sum_{l=1}^{M} \tilde{s}_l Q_{il}(u) - Q_i^0(u),$$
$$s_j = H(\sum_{k=1}^{n} w_{jk} u_k - h_j), \quad \tilde{s}_l = H(\sum_{k=1}^{M} \tilde{w}_{lk} u_k - \tilde{h}_l), \tag{2.2}$$

where H is the unit step function $H(y) = 1, y \geq 0$, and $H(y) = 0, y < 0$, $P_{ik}, P_i^0, Q_{il}, Q_i^0$ are positive, smooth functions of u representing production, basal production, consumption, and basal consumption, respectively. Here $w, \tilde{w}$ are matrices describing the interactions between the u variables, $i = 1, 2, ..., n$, $j = 1, 2, ..., N$, $l = 1, ..., M$ and $h, \tilde{h}$ are thresholds.

One will usually look for solutions of the piecewise-smooth dynamics (2.2) such that trajectories of $\boldsymbol{u}$ are continuous. However, we can easily extend the above definitions in order to cope with jumps of the continuous variables. Similarly to impact systems occurring in mechanics [DB08], the jumps of the continuous variables can be commanded by the following rule: $\boldsymbol{u}$ instantly changes to $\boldsymbol{p}_j^{\pm}(\boldsymbol{u})$ whenever a discrete variable $\hat{s}_j = H(\sum_{k=1}^n \hat{w}_{jk} u_k - \hat{h}_j)$ changes. The $\pm$ superscripts correspond to changes of $\hat{s}_j$ from 0 to 1 and from 1 to 0, respectively. We can consider reversible jumps in which case the functions $\boldsymbol{p}_j^{\pm}(\boldsymbol{u})$ satisfy $\boldsymbol{p}^+ \circ \boldsymbol{p}^- = Id$. The typical example in molecular biology is the cell cycle. In this case, the command to divide at the end of mitosis is irreversible and corresponds to $\boldsymbol{p}_j^+(\boldsymbol{u}) = \boldsymbol{u}/2$. No return is possible, $\boldsymbol{p}_j^-(\boldsymbol{u}) = \boldsymbol{u}$.

The class of models (2.2) is too general. We will restrict ourselves to a subclass of piecewise smooth systems where smooth production and degradation terms are assumed multivariate monomials in u, plus some basal terms:

$$\begin{aligned} P_{ik}(\mathbf{u}) &= a_{ik} u_1^{\alpha_1^{ik}} \dots u_n^{\alpha_n^{ik}}, \\ P_i^0(\mathbf{u}) &= a_i^0 \\ Q_{il}(\mathbf{u}) &= \tilde{a}_{il} u_1^{\tilde{\alpha}_1^{il}} \dots u_n^{\tilde{\alpha}_n^{il}} \\ Q_i^0(\mathbf{u}) &= \tilde{a}_i^0 u_i \end{aligned} \tag{2.3}$$

which will be chosen according to an heuristic presented in the next sections.

This restriction does not reduce the power of the method. As argued in the introduction, the monomial rates represent good approximations for nonlinear networks of biochemical reactions with multiple separated timescales [RGZL08, GR08]. More generally, rational functions are good candidates for general rate laws [LUK10]. However, when concentrations are very large or very small the monomial laws are recovered. For instance, Michaelis Menten, Hill, or Goldbeter-Koshland reactions switch from a saturated regime where rates are constant to a small concentration regime where rates follow power laws. Finally, by methods described in [Vak02, VG03] one can show that the above subclass of models can approximate with arbitrary precision any structurally stable dynamics.

These models have several advantages with respect to standard models in molecular biology and neuroscience based on differential equations. They allow us to simulate, in a fairly simple manner, discontinuous transitions occurring in such systems (see a typical graph describing time evolution of protein concentration within cellular cell cycle, Fig.1). The discontinuous transitions result either from fast processes or from strongly non-linear (thresholding) phenomena. This class of models is also scalable in the sense that more and more details can be introduced at relatively low cost, by increasing the number of discrete variables and the size of the interaction matrices.

The definition of the rates slightly extends the one used in S-systems, introduced by Savageau [SV87]. Our choice was motivated by the fact that S-systems proved their utility as models for metabolic networks whose dynamics we want to encompass by considering the modes. The introduction of basal terms avoids spurious long living states when some products have zero concentrations.

3 Regulated Reaction Graphs and Hybrid Reaction Schemes

Interaction mechanisms in molecular biology can be schematized as regulated reaction graphs.

A regulated reaction graph is a quadruple (V, R, E, E_r). The triplet (V, R, E), where $E \subset V \times R \cup R \times V$, defines a reaction bipartite graph, ie $(x, y) \in E$ iff $x \in V, y \in R$ and x is a substrate of R, or $x \in R, y \in V$ and y is a product of x. $E_r \subset V \times R$ is the set that defines regulations, $(x, z) \in E_r$ if the rate of the reaction $z \in R$ depends on $x \in V$ and x is not a substrate of R.

Similar structures of regulated reactions where proposed elsewhere for non-hybrid models [LUK10].

In order to define a hybrid model we first need a *hybrid reaction scheme.* This consists in saying, for each given species, whether its production/degradation can be switched on and off and by which species, also which species modulate the production/degradation of a given species in a smooth way. This means specifying a partition of the regulations $E_r = E_r^d \cup E_r^c$. A regulation $(x, r) \in E_r^d$ is discrete if the decision to switch on and off the reaction r depends (among others) on x. Discrete interactions manifest themselves punctually as a consequence of thresholding and/or of rapid phenomena. The continuous regulations guide the dynamics of the modes. Similarly, there is a partition of the reactions $R = R^s \cup R^c$. A reaction r belongs to the switched reactions $r \in R^s$ if $(x, r) \in E_r^d$, for some $x \in V$. The role of the regulators (continuous if they modulate the reaction rate, discrete if they contribute to switching it on and off) should be indicated on the graph together with the signs of the regulations.

4 Identification of Piecewise Smooth Models

We would like to develop methods allowing to find the parameters of a model from the class introduced above that best describes the observed trajectories of a biological system. These trajectories can come from experiments or can be produced by non-hybrid models. In both situations we obtain a model whose parameters can be easily interpreted in biological terms. The hybrid model can be further analyzed or used to model more complex situations.

In the following we present a reverse engineering algorithm that works well for systems with sharp transitions.

Data. n trajectories (time series) $u_1(t), ..., u_n(t)$ given at time moments $t_0, t_1, ..., t_N$. A regulated reaction graph (the smooth/discrete partition of the regulations can be unspecified).

Output. A model of the type (2.2), (2.3) with values of the parameters that fit well the data.

The algorithm has several steps, some of them involving several alternative numerical solutions. For some of the steps the choice of the numerical solution was adapted to the application presented in the paper, which is the reconstruction of a hybrid cell cycle oscillator.

I. Choice of hybrid reaction scheme and of monomials giving the smooth part of the rates.
The reaction rates have the forms given by (2.3). The monomial exponents α_{ij}, $\tilde{\alpha}^i_j$, the rate functions defining the modes, the mode switching and the jumps can be obtained from the following heuristic rules:

i) If a reaction j is activated then $\alpha^i_j = 1$ for all activators and $\alpha^i_j = -1$ for all inhibitors i in the absence of cooperativity. Cooperativity may be taken into account by considering $|\alpha^i_j| > 1$.
ii) Basal rates are constant for reactions without substrates and proportional to the concentration of the substrate otherwise.
iii) If activated reactions are present with intermittence, their non-basal rates are multiplied by discrete variables s_i; this defines the mode switching.
iv) If a continuous variable u_i is known to induce a jump decision (for instance cell division), it should appear in the definition of the jump discrete variables $\hat{s}$. The functions $\mathbf{p}(\mathbf{u})$ follow from biological observations.

Once the hybrid reaction scheme chosen, we want to fit the remaining model parameters in order to reproduce the observed dynamics.

II. Detection of the events locations.
We look for K time intervals $I_1, I_2, ..., I_K$. The dynamics on each of the intervals is smooth, it is given by (2.2) with the s variables fixed. Mode transitions (change of the variables) occur at the borders of these intervals. We denote the switching times as $\tau_1, ...\tau_K$.

Finding τ_k is a problem of singularity detection. This could be done by various methods, for example by wavelet analysis. Here we decided to use the derivatives of the reaction rates to locate the mode switching events. The peaks of these derivatives indicate the positions of switching events, whereas the sign of the derivatives indicate the sign of the change (activation if positive, inactivation if negative). With this simple criterion we are able to reconstruct the sequence of modes which is defined by the values of the boolean variables $s(t)$.

III. Determining the mode internal parameters.
The previous steps define a set of modes and the static event location. Given a choice of the modes internal parameters the hybrid trajectories can be integrated without knowing the discrete regulations (this will allow the dynamic event location at the next step): the values of s between two successive events are enough. Modes internal parameters are obtained by optimization. Let $u_i^{modes}(t)$ be the continuous hybrid trajectories obtained by integrating the modes between the calculated transition times. We use a parallel version of Lam's simulated annealing algorithm [CDR99] to minimize the following objective function:

$$F = \sum_{i,k} C_k (u_i^{modes}(t_k) - u_i(t_k))^2,$$

where C_k are positive weights. The choice of the weights depends on the dynamical features one wants to reproduce. For instance, for the cell cycle application we choose weights that increase with time. We thus penalize large time deviations that can arise from the loss of synchronicity among variables u_i and avoid the period misfit that could arise between the hybrid and the smooth dynamics after dynamic event location.

IV. Determining the mode control parameters and dynamic event location. Let $s_m = H(\sum_{(m,j)\in E^r} w_{mj}u_j - h_j)$ be the discrete variables determined above. Let s_k^m be the constant values of s_m on I_k. Consider now the optimal trajectories $u_i^{modes*}(t_l)$ obtained before.

Then, one should have

$$(\sum_{(m,j)\in E^r} w_{mj}u_j^{modes*}(t_l) - h_j)s_k^m > 0, \text{for all}\, t_l \in I_k, \tag{4.1}$$

which is a linear programming problem for w_{mj} that can be resolved (if it has a solution) in polynomial time.

5 Examples

A simple cell cycle model. As a simple example let us consider the minimal model proposed by Tyson for Cdc2 and Cyclin interactions [Tys91]. This model, which contains initially 6 species and 9 reactions, can be reduced to only 2 species and 4 reactions (details of the reduction will be given elsewhere), while keeping the same dynamics. The two species left are Cyclin-Cdk complexes, with two phosphorylation states: phosphorylation of both monomers ($Cpp := Cyclin_p.Cdc2_p$), or only Cyclin phosphorylated ($Cp := Cyclin_p.Cdc2$).

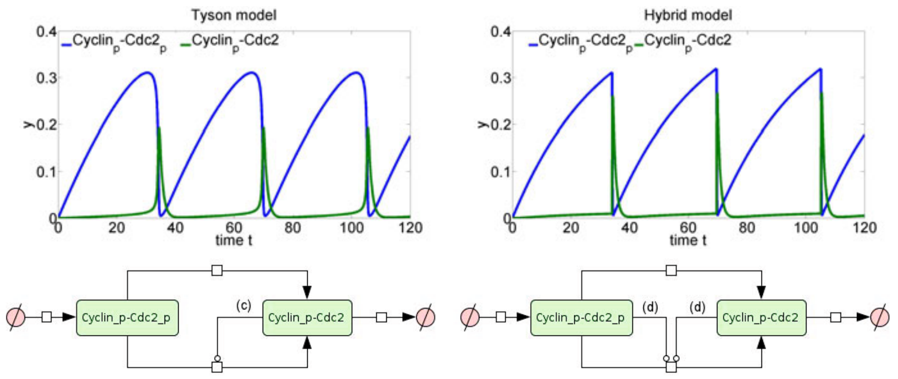


Fig. 1. (Top Left) Trajectories of the non-hybrid model by Tyson [Tys91]. (Top Right) Trajectories of the hybrid model. (Bottom Left) Reaction graph of the non-hybrid model. (Bottom Right) Reaction graph of the hybrid model.

$$\begin{aligned}\frac{d[Cpp]}{dt} &= k_1 - k_4'[Cpp] - k_4[Cpp][Cp]^2 \\ \frac{d[Cp]}{dt} &= -k_6[Cp] + k_4'[Cpp] + k_4[Cpp][Cp]^2\end{aligned} \tag{5.1}$$

The regulated reaction graph and the hybrid scheme are represented in Fig.1. The dynamics of this model is quite simple. The linear dephosphorylation is slower than the production of Cpp. This create an accumulation of Cpp. Then at some threshold the Cp produced activates the second, faster, dephosphorylation, which drains the accumulated Cpp. Here we model this second, faster reaction as an hybrid reaction, totally controlled by thresholds. This is justified by the observed peaks of the rate derivative (Fig.2) and leads to the following hybrid model:

$$\begin{aligned}\frac{d[Cpp]}{dt} &= \tilde{k}_1 - \tilde{k}_4'[Cpp] - \tilde{k}_4 s[Cpp] \\ \frac{d[Cp]}{dt} &= -\tilde{k}_6[Cp] + \tilde{k}_4'[Cpp] + \tilde{k}_4 s[Cpp]\end{aligned} \tag{5.2}$$

where $s = H(w_1[Cpp] + w_2[Cp] - h)$ is the boolean variable.

After the parameter fit we find that w_1 and w_2 are both positive.

For this model no jumps of the continuous variables are needed. Indeed, at the end of mitosis, all continuous variables have small values. Bringing their values to half would not change much the behavior of the model.

Generic mammalian cell cycle model. A complex example with mode switching and jumps is obtained from Novak and Tyson 2006 generic cell cycle model [Csi06]. The generic model contains four different versions, each of them reproduce the cell cycle for a different eukaryote organism : Mammalians, Xenopus embryo, Budding yeast, Fission Yeast. Our study will only focus on the mammalian version. This model contains 12 species and 34 reactions.

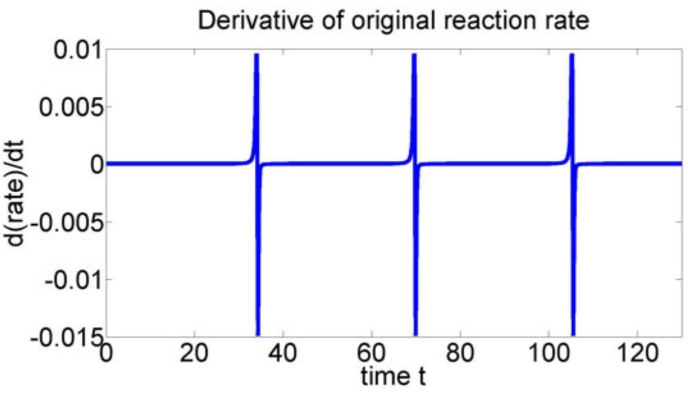


Fig. 2. Rate derivative for the simple cell cycle model

We briefly discuss the steps of the algorithm applied to this model.

Choice of the hybrid scheme. Four of these reactions are typically switch-like, following Goldbeter-Koshland kinetics. Another reaction is following Hill kinetics, and also shows switch-like behavior. These reactions are replaced by switched reactions whose rates are simplified monomial rate multiplied by a boolean variable.

For instance the reaction that produces Cyclin-B, induced by the cell mass, is defined by the following kinetic rate:

$$\mathrm{R} = \mathrm{ksb}_{pp}\,[\mathrm{Mass}]\,\mathrm{GK}(\mathrm{kafb}\,[\mathrm{CycB}], \mathrm{kifb}, \mathrm{Jafb}, \mathrm{Jifb}) \tag{5.3}$$

In this case we can replace the Goldbeter-Koshland (GK) function by a boolean variable (see Fig.3) and obtain the following simpler rate:

$$\mathrm{R}' = \mathrm{ksb}_{pp}\,[\mathrm{Mass}]\,\mathrm{s} \tag{5.4}$$

where s is a boolean variable.

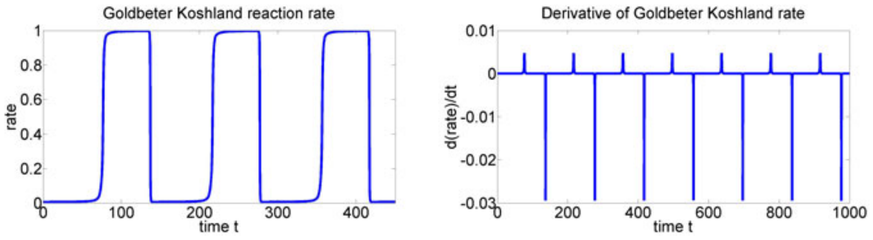


Fig. 3. left: Reaction flow of the GK function, right: Derivative of reaction flow of the GK function

Detection of the transitions. Static event locations follow from the positions of the peaks of the derivative of the reactions rates with respect to time. For the reaction considered above, we find 2 peaks per period of the cell cycle model (see Fig. 3), which correspond to switching the reaction from an active state to an inactive one and back. At the end of this step, we obtain a list of the transitions points for each switched reaction, and the status of the switched reactions during each mode (ie between successive transitions) as boolean variables.

Fitting the hybrid model parameters. Once we have this definition, we can fit the model, using the parallel simulated annealing algorithm. As a control we can see the results of the fitting on the rate of the reaction considered above (Fig.4).

Computing the mode control parameters. The final goal is to obtain a dynamic definition of these events location, by computing the regulation matrix w and the thresholds h for all the boolean variables.

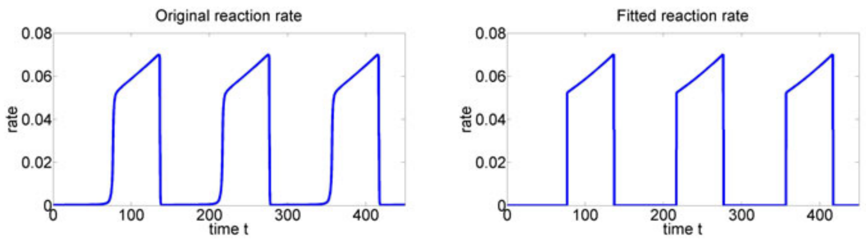


Fig. 4. left: Original reaction flow, right: Hybrid reaction flow

For instance, the reaction discussed is controlled by the function $f(u) = -37.32[CycA] - 1.694[CycB] - 173.7[CycE] - 177.8[APCP] + 331.2[Cdc20a] + 97.8[Cdc20i] - 78.5[Cdh1] - 107.1[CKI] + 0.3481 Mass + 53.52[pB] + 1337[TriA] + 39.57[TriE]$, which is the parameter of the Heaviside function. The sign of this event function will control the state of the reaction, keeping it inactive if the function is negative, and activating the reaction when the function is positive (Fig.5).

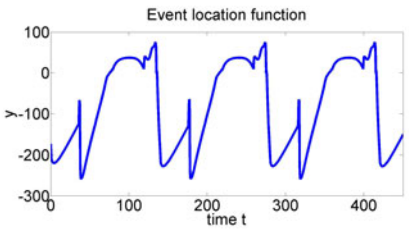


Fig. 5. Event location function, ie the Heaviside function parameter

On Fig.6, we can see the trajectories of the hybrid model (with dynamic event location), compared to those of the original model.

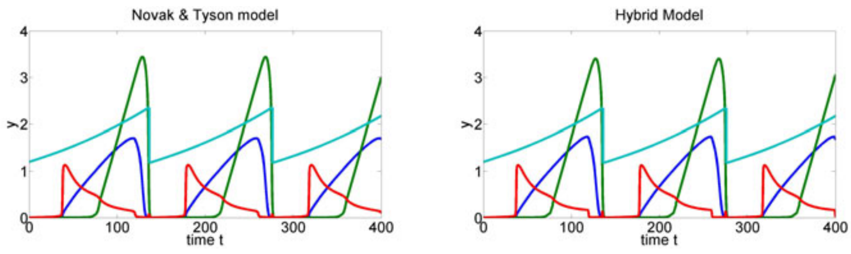


Fig. 6. (Left) Trajectories of four main variables of the non-hybrid model by Novak and Tyson [NT04]. (Right) Trajectories of the hybrid model. (blue : Cyclin-A, green : Cyclin-B, red : Cyclin-C, aqua : cell size).

6 Conclusion

The results that we present are a proof of principle that piecewise smooth hybrid models can be constructed with a simple heuristic from basic information about biochemical interactions. Using this class of hybrid models instead of piecewise-linear approximations provides, in many situations, a better balance between discrete and smooth interactions. The identification algorithm proposed in the paper combines the static location of the events, the identification of the modes by simulated annealing, and the identification of the mode control parameters by dynamic location. The hardest step of this algorithm is the simulated annealing. Furthermore, for large models, we expect several solutions for the mode control parameters. We are currently improving the algorithm to cope with these situations. A better choice of the modes dictated by model reduction techniques could reduce the time for simulated annealing. Also, we are investigating the use of event location functions that are linear in the logarithms of the continuous variables. According to the ideas of the introduction, these nonlinear location functions will indicate changes of the dominant monomials in the rate functions, more accurately than the linear location functions. Moreover, they can be obtained directly from the initial smooth model without the need to solve (eventually undetermined) dynamic location inequations. Improved segmentation techniques are needed for future application of the algorithm directly to data.

In the future we will apply the heuristic and the fitting algorithm to model complex situations when signaling pathways interact with the eucaryotic cell cycle. The resulting hybrid models will also be used to investigate emerging properties of regulatory networks such as viability and robustness.

Acknowledgements. We thanks J.Reinitz for sending us a version of his parallel simulated annealing code.

References

[BBG00] Barton, P.I., Banga, J.R., Galan, S.: Optimization of hybrid discrete/continuous dynamic systems. Computers & Chemical Engineering 24(9-10), 2171–2182 (2000)

[CDR99] Chu, K.W., Deng, Y., Reinitz, J.: Parallel simulated annealing by mixing of states 1. Journal of Computational Physics 148(2), 646–662 (1999)

[Csi06] Csikász-Nagy, A., Battogtokh, D., Chen, K.C., Novák, B., Tyson, J.J.: Analysis of a generic model of eukaryotic cell-cycle regulation. Biophysical Journal 90(12), 4361–4379 (2006)

[DB08] Di Bernardo, M.: Piecewise-smooth dynamical systems: theory and applications. Springer, Heidelberg (2008)

[DJGH$^+$04] De Jong, H., Gouzé, J.L., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulation of genetic regulatory networks using piecewise-linear models. Bulletin of Mathematical Biology 66(2), 301–340 (2004)

[DMT10] Dang, T., Maler, O., Testylier, R.: Accurate hybridization of nonlinear systems. In: Proceedings of the 13th ACM International Conference on Hybrid systems: Computation and Control, pp. 11–20. ACM, New York (2010)

[FA88] Filippov, A.F., Arscott, F.M.: Differential equations with discontinuous righthand sides. Springer, Heidelberg (1988)

[GR08] Gorban, A.N., Radulescu, O.: Dynamic and static limitation in reaction networks, revisited. In: West, D., Marin, G.B., Yablonsky, G.S. (eds.) Advances in Chemical Engineering - Mathematics in Chemical Kinetics and Engineering. Advances in Chemical Engineering, vol. 34, pp. 103–173. Elsevier, Amsterdam (2008)

[GRW07] Gebert, J., Radde, N., Weber, G.W.: Modeling gene regulatory networks with piecewise linear differential equations. European Journal of Operational Research 181(3), 1148–1165 (2007)

[GRZ10] Gorban, A.N., Radulescu, O., Zinovyev, A.Y.: Asymptotology of chemical reaction networks. Chemical Engineering Science 65, 2310–2324 (2010)

[LUK10] Liebermeister, W., Uhlendorf, J., Klipp, E.: Modular rate laws for enzymatic reactions: thermodynamics, elasticities and implementation. Bioinformatics 26(12), 1528 (2010)

[MS00] Matveev, A.S., Savkin, A.V.: Qualitative theory of hybrid dynamical systems. Birkhäuser, Basel (2000)

[NT04] Novak, B., Tyson, J.J.: A model for restriction point control of the mammalian cell cycle. Journal of Theoretical Biology 230(4), 563–579 (2004)

[RGZL08] Radulescu, O., Gorban, A.N., Zinovyev, A., Lilienbaum, A.: Robust simplifications of multiscale biochemical networks. BMC Systems Biology 2(1), 86 (2008)

[SV87] Savageau, M.A., Voit, E.O.: Recasting nonlinear differential equations as S-systems: a canonical nonlinear form. Mathematical Biosciences 87(1), 83–115 (1987)

[SWM$^+$07] Shorten, R., Wirth, F., Mason, O., Wulff, K., King, C.: Stability Criteria for Switched and Hybrid Systems. SIAM Review-The Flagship Journal of the Society for Industrial and Applied Mathematics 49(4), 545–592 (2007)

[Tav87] Tavernini, L.: Differential automata and their discrete simulators. Nonlinear Anal. Theory Methods Applic. 11(6), 665–683 (1987)

[Tys91] Tyson, J.J.: Modeling the cell division cycle: cdc2 and cyclin interactions. Proceedings of the National Academy of Sciences of the United States of America 88(16), 7328 (1991)

[Vak02] Vakulenko, S.: Complexité dynamique des réseaux de Hopfield: Dynamical complexity of the Hopfield networks. Comptes Rendus Mathematique 335(7), 639–642 (2002)

[VG03] Vakulenko, S., Grigoriev, D.: Complexity of gene circuits, pfaffian functions and the morphogenesis problem. Comptes Rendus Mathematique 337(11), 721–724 (2003)

Parsimonious Reconstruction of Network Evolution

Rob Patro[1,2], Emre Sefer[1,2], Justin Malin[1,3], Guillaume Marçais[1,4], Saket Navlakha[5], and Carl Kingsford[1,2,3,4]

[1] Center for Bioinformatics and Computational Biology
[2] Department of Computer Science
[3] Computational Biology, Bioinformatics and Genomics Concentration, Biological Sciences Graduate Program
[4] Program in Applied Mathematics, Statistics, and Scientific Computation
University of Maryland, College Park, MD 20742, USA
[5] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. We consider the problem of reconstructing a maximally parsimonious history of network evolution under models that support gene duplication and loss and independent interaction gain and loss. We introduce a combinatorial framework for encoding network histories, and we give a fast procedure that, given a set of duplication histories, in practice finds network histories with close to the minimum number of interaction gain or loss events. In contrast to previous studies, our method does not require knowing the relative ordering of unrelated duplication events. Results on simulated histories suggest that common ancestral networks can be accurately reconstructed using this parsimony approach.

1 Introduction

High-throughput experiments have revealed thousands of regulatory and protein-protein interactions that occur in the cells of present-day species. To understand why these interactions take place, it is necessary to view them from an evolutionary perspective. In analogy with ancestral genome reconstruction [22], we consider the problem of predicting the topology of the common ancestor of pathways, complexes, or regulatory programs present in multiple extant species.

Generating plausible ancestral networks can help answer many natural questions that arise about how present-day networks have evolved. For example, joint histories can be used to compare the conservation and the route to divergence of corresponding processes in two species. This allows us to more finely quantify how modularity has changed over time [15] and how interactions within a protein complex may have reconfigured across species starting from a single shared state [24]. Such analysis can also be integrated to develop better network alignment algorithms and better network-based phylogenies [11,27,8,9,16], and it can be used to study robustness and evolvability [1,10,26]. Further, inferred changes in metabolic networks can be linked to changes in the biochemical environment in which each species has evolved, and this can reveal novel mechanisms of ecological adaptation [4,3]. Finally, comparing network histories inferred using different model parameters can be used to estimate the likelihoods of various evolutionary events [18,21].

There has been some recent work on reconstructing ancestral interactions. Gibson and Goldberg [13] presented a framework for estimating ancestral protein interaction networks that handles gene duplication and interaction loss using gene trees reconciled against a species phylogeny. However, their approach assumes that interaction losses occur immediately after duplication and does not support interaction gain outside of gene duplication. These assumptions are limiting because interaction loses may occur well after duplication, and independent gains are believed to occur at non-trivial rates [17]. Dutkowski and Tiuryn [8] provided a probabilistic method for inferring ancestral interactions with the goal of improved network alignment. Their approach is based on constructing a Bayesian network with a tree topology where binary random variables represent existence or non-existence of potential interactions. A similar graphical model was proposed by Pinney et al. [25], who applied it to inferring ancestral interactions between bZIP proteins. In the former method, interaction addition and deletion is assumed to occur only immediately following a duplication or speciation event. Further, both methods assume the relative ordering of duplication events is known even between events in unrelated homology groups. Pinney et al. [25] also explore a parsimony-based approach [19] and find it to work well; however, it too assumes a known ordering of unrelated duplication events. The main drawback of these approaches is that the assumed ordering comes from sequence-derived branch lengths, which do not necessarily agree with rates that would be estimated based on network evolution [31]. This motivates an approach such as we describe below that does not use branch lengths as input.

Zhang and Moret [31,30] use a maximal likelihood method to reconstruct ancestral regulatory networks as a means to improve estimation of regulatory networks in extant species. Mithani et al. [20] study the evolution of metabolic networks, but they only model the gain and loss of interactions amongst a fixed set of metabolites, whereas we also consider node duplication and loss encoded by a tree. Navlakha and Kingsford [21] present greedy algorithms for finding high-likelihood ancestral networks under several assumed models of network growth. They applied these methods to a yeast protein interaction network and a social network to estimate relative arrival times of nodes and interactions and found that the inferred histories matched many independently studied properties of network growth. This attests to the feasibility of using networks to study evolution. The authors, however, only consider a single network at a time, and there is no guarantee that independent reconstruction of two networks will converge to a common ancestor.

Here, we introduce a combinatorial framework for representing histories of network evolution that can encode gene duplication, gene loss, interaction gain and interaction loss at arbitrary times and does not assume a known total ordering of duplication events. We show that nearly-minimal parsimonious histories of interaction gain and loss can be computed in practice quickly given a duplication history. In simulated settings, we show that these parsimonious histories can be used to accurately reconstruct a common ancestral regulatory network of two extant regulatory networks.

2 A Framework for Representing Network Histories

Any natural model of network evolution will include events for gene duplication, gene loss, interaction gain, and interaction loss. Many such growth models have been studied (e.g. [6,29,23,14,1,30]). We now describe how these events can be encoded in a history graph.

Consider a set V of proteins or genes (henceforth "nodes") descended from a common ancestor by duplication events. Those duplication events can be encoded in a binary *duplication tree* T with the items of V as the leaves. An internal node u in T represents a duplication event of u into its left and right children, u_L and u_R. In this representation, after a duplication event, the node represented by u conceptually does not exist anymore and has been replaced by its two children. The leaves of a duplication tree are labeled *Present* or *Absent*. Absent leaves represent products of duplication events that were subsequently lost. A collection of such trees is a *duplication forest* F.

The gain and loss of interactions can be represented with additional non-tree edges placed on a duplication forest. A non-tree edge $\{u, v\}$ represents an *edge flip event*, where the present / absent state of the interaction between u and v is changed to Present if the interaction is currently Absent or to Absent if the interaction is currently Present. Let P_u and P_v be the paths from nodes u and v to the root. An interaction exists between u and v if there are an odd number of such flip non-tree edges between nodes in P_u and P_v. Every non-tree edge between P_u and P_v, therefore, represents alternatively interaction creation or deletion between nodes u and v in the evolution of the biological network.

A graph H consisting of the union of a duplication forest and flip non-tree edges is a *network history*. A history H *constructs* a graph G when the Present leaves of the duplication forest in H correspond to the nodes of G and the flip edges of H imply an interaction between u and v if and only if $\{u, v\}$ is an interaction in G. See Figure 1 for an example history.

Not all placements of non-tree edges lead to a valid network history. The interaction histories have to be consistent with some temporal embedding of the tree. Let t_u^c and t_u^d be respectively the time of creation and duplication of node u, Naturally, $t_u^c < t_u^d$, $t_u^d = \infty$ if u is a Present leaf, and if v is the child of u, then by definition we have

$$t_u^c < t_u^d = t_v^c < t_v^d. \tag{1}$$

If $\{u, v\}$ is a flip edge, then the time $t_{\{u,v\}}$ of appearance of this edge must satisfy

$$t_u^c \leq t_{\{u,v\}} < t_u^d \quad \text{and} \quad t_v^c \leq t_{\{u,v\}} < t_v^d, \tag{2}$$

because an event between u and v can only occur when both u and v exist. A history graph H is said to be *valid* if there exist t_u^c, t_u^d for every node u such that conditions (1) and (2) are satisfied for every non-tree edge.

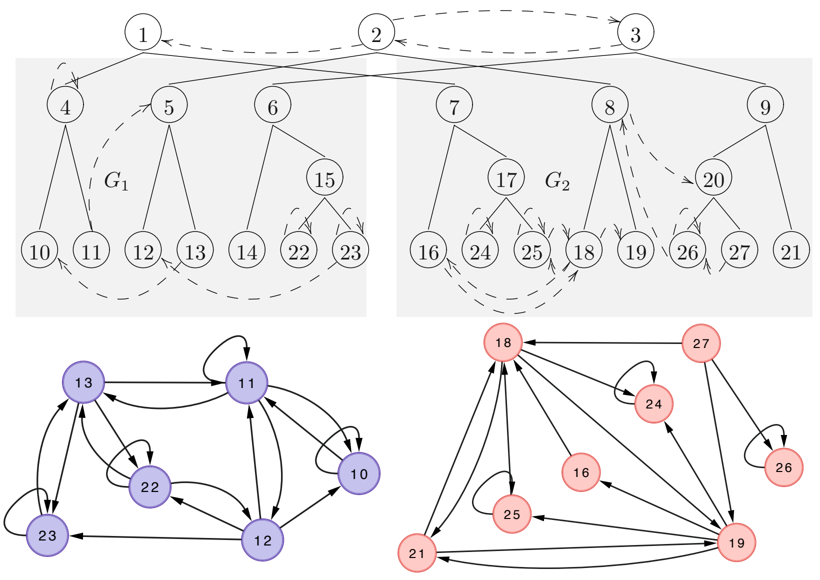


Fig. 1. A duplication forest (solid edges at top) with the non-tree edges (dashed) necessary to construct G_1 and G_2 (shown at bottom). Nodes 1, 2, and 3 represent the 3 homology groups present in the ancestral graph. Node 14 was lost. As an example of the connectivity induced by the non-tree edges, consider edge $(27, 18)$ in G_2 which is implied by the directed non-tree edge from $(3, 2)$. However, the reverse edge, $(18, 27)$, which is implied by $(2, 3)$, does not exist because its state is flipped by $(8, 20)$.

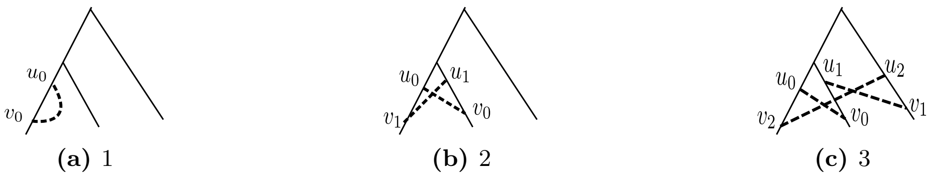


Fig. 2. Blocking loops of size 1, 2 and 3. The solid lines represent a subset of the tree T. The dashed lines are non-tree edges representing interaction flip events.

Whether a particular history is valid can be checked combinatorially using the following alternative characterization of validity. A *k-blocking loop* is a set of flip edges $\{\{u_i, v_i\}\}_{0 \le i < k}$ such that u_{i+1} is an ancestor of v_i in the tree for $0 \le i < k$ (where the index $i + 1$ is taken modulo k). See Figure 2 for examples. Blocking loops are not permitted in valid histories and, conversely, the non-existence of blocking loops implies that a history is valid, as shown in Prop. 1.

Proposition 1. *A history graph H is valid if and only if it does not have any blocking loop of any length.*

Proof. Suppose there is a k-blocking loop. Using the same notation as above, we have the inequalities

$$t^d_{u_0} > t_{\{u_0,v_0\}} \geq t^c_{v_0} \geq t^d_{u_1} > t_{\{u_1,v_1\}} \geq \ldots \geq t^c_{v_{k-1}} \geq t^d_{u_0},$$

which is a contradiction. Hence, to not have any blocking loops is necessary.

Conversely, suppose that H does not have any blocking loops. We assign times to the nodes and non-tree edges using a modified depth-first search (DFS) algorithm following the tree edges only. First, the root of the tree is given a creation time of 0. During DFS, just before calling DFS recursively on the left and right children of a node u, we set the duplication time $t^d_u = \max\{\max t_{\{u,v\}} + 1, t^c_u + 1\}$, where the second max is taken over all non-tree edges adjacent to u. Also, we set the creation time of the children $t^c_{u_L} = t^c_{u_R} = t^d_u$.

When DFS visits a node u with some edge $\{u, v\}$ where v has not been assigned a creation time, u is added to a set Q and DFS is not called recursively on the children of u. The main loop consists of calling DFS again on all the nodes in Q until this set is empty. By construction, the algorithm assigns times which satisfy conditions (1) and (2). Therefore, if the algorithm terminates, H is a valid history.

At each main iteration, the nodes in the set Q are all the nodes u for which t^c_u is set but t^d_u is not set. It suffices to show that at each such iteration, at least one of the nodes in the set Q will not be added again to Q by a call to DFS. In other words, for at least one node $u \in Q$, every non-tree edge $\{u, v\}$ has t^c_v set. For a contradiction, suppose not. Take $u_1 \in Q$ and $\{u_1, v_1\}$ with $t^c_{v_1}$ not set. There is necessarily an ancestor of v_1, call it u_2, which is in Q. Similarly, take $\{u_2, v_2\}$ with $t^c_{v_2}$ not set and its ancestor $u_3 \in Q$, and so on. Because Q is finite, $u_j = u_i$ for some $j > i$, and we constructed a blocking loop. Hence, the algorithm must terminate. □

3 Parsimonious Reconstruction of a Network History

Traditional phylogenetic inference algorithms and reconciliation between gene and species trees can be used to obtain duplication and speciation histories [5,7,2]. What remains is the reconstruction of interaction gain and loss events. This leads to the following problem:

Problem 1. Given a duplication forest F and an extant network G, find H, a valid history constructing G, with a minimum number of flip edges.

We will show that nearly optimal solutions to this problem for a large range of instances can be solved in polynomial time in practice. Whether Problem 1 is NP-hard or admits a polynomial-time algorithm for all instances remains open.

3.1 A Fast Heuristic Algorithm

The challenge of Problem 1 comes from avoiding the creation of blocking loops. A polynomial-time algorithm can find a minimum set of flip edges that reconstructs a graph G and does not contain 1- and 2-blocking loops but allows longer blocking loops. We define an *interaction encoding* of $G = (V, E)$ as a function $f_G : V \times V \to \{0, 1\}$ such that: $f_G(u, v) = 1$ if $\{u, v\}$ is an interaction in G and $f_G(u, v) = 0$ otherwise. We omit the subscript on f_G if G is clear from the context.

The following intertwined dynamic programming recurrences find the minimum number of flip edges required for H to construct a given graph G if blocking loops of length ≥ 3 are allowed. First, $S(u, f)$ finds the minimum number of flip edges for the subtree rooted at u and interaction encoding f:

$$S(u, f) = S(u_L, f) + S(u_R, f) + A(u_L, u_R, f). \tag{3}$$

The expression $A(u, v, f)$ gives the minimum number of flip edges that should be placed between the subtree rooted at u and the subtree rooted at v. This can be computed using the recurrence:

$$A(u, v, f) = \min \begin{cases} A(u_L, v, f) + A(u_R, v, f) \\ A(u, v_L, f) + A(u, v_R, f) \\ 1 + A(u_L, v, \bar{f}) + A(u_R, v, \bar{f}) \\ 1 + A(u, v_L, \bar{f}) + A(u, v_R, \bar{f}). \end{cases} \tag{4}$$

In the above, if one of u or v is a leaf but the other is not, the options that look at non-existent children are disallowed.

The function $\bar{f}$ in Eqn. (4) is defined as $1 - f$ and thus represents a function such that $\bar{f}(x)$ has opposite parity from $f(x)$ for all x. The A recurrence considers two possible options: (1) We connect u and v with a non-tree edge, this costs us 1 and flips the parity of all interactions going between the subtree rooted at u and the subtree rooted at v; or (2) We do not connect u and v with a flip edge. This costs 0 and keeps the parity requirement the same. Regardless of the choice to create an edge, since we are not allowed to have a 2-blocking loop, either (a) we possibly connect u to some descendant of v (and do not connect v to a descendant of u) or (b) we possibly connect v to some descendant of u (and do not connect u to a descendant of v).

The base case for the S recurrence when u is a leaf and the base case for the A recurrence when u and v are leaves are:

$$S(u, f) = 0 \quad \text{and} \quad A(u, v, f) = f(u, v).$$

The minimum number of flip edges needed to turn a duplication forest F into a history constructing G (allowing blocking loops of ≥ 3) is then given by $\sum_r S(r, d_G) + \sum_{r,q} A(r, q, d_G)$, where d_G is the interaction encoding of G, and the sums are over roots r, q of the trees in F. Standard backtracking can be used to recover the actual minimum edge set. The dynamic program runs in $O(n^2)$ time and space because only two functions f are ever considered: d_G, and $\bar{d_G}$. This yields $\approx n \times n \times 2$ subproblems, each of which can be solved in constant time.

3.2 Removing Blocking Loops

If the solution contains blocking loops of length ≥ 3, one can choose an edge in some blocking loop, forbid that edge from appearing in the solution, and rerun the dynamic program. Because there are $O(n^2)$ possible non-tree edges, iterating this procedure will terminate in polynomial time. In practice, we can choose to exclude the non-tree edge that participates in the largest number of loops. We repeat this until a valid solution is obtained. In the worst case, one may obtain a solution where all non-tree edges are placed at leaves, but in practice long blocking loops do not often arise, and the obtained solutions are close to optimal (see Sec. 4.2).

3.3 Reconstruction of a Common Ancestor of Two Graphs

Given extant networks of several species, in addition to the reconstructed history, we seek a parsimonious estimate for their common ancestor network. Specifically, given extant networks G_1 and G_2, with interaction encodings d_1 and d_2, and their duplication forests F_1 and F_2, we want to find an ancestral network $X = (V_X, E_X)$ such that the cost of X evolving into G_1 and G_2 after speciation is minimized. V_X is the set of roots of the homology forests. We assume that the networks of the two species evolved independently after speciation. Therefore, we can use the recurrence above applied to F_1 and F_2 to compute $A_{F_1}(r, q, d_1)$ and $A_{F_2}(r, q, d_2)$ independently for $r, q \in V_X$, and then select interactions in X as follows. E_X of X is given by the pairs $r, q \in V_X \times V_X$ for which creating an interaction leads to a lower total cost than not creating an interaction. Formally, we place an interaction $\{r, q\}$ in E_X if

$$1 + A_{F_1}(r, q, \bar{d_1}) + A_{F_2}(r, q, \bar{d_2}) < A_{F_1}(r, q, d_1) + A_{F_2}(r, q, d_2). \quad (5)$$

Rule (5) creates an interaction in X if doing so causes the cost of parsimonious histories inferred for G_1 and G_2 between the homology groups associated with r and q to be smaller than if no interaction was created.

3.4 Modifications for Self-loops

Self-loops (homodimers) can be accommodated by modifying recurrence (3):

$$S'(u, f) = \begin{cases} S'(u_L, f) + S'(u_R, f) + A(u_L, u_R, f) \\ 1 + S'(u_L, \bar{f}) + S'(u_R, \bar{f}) + A(u_L, u_R, \bar{f}). \end{cases} \quad (6)$$

The intuition here is that paying cost 1 to create a self-loop on node u creates (or removes) interactions, including self-loops, among all the descendants of u.

3.5 Modifications for Directed Graphs

Finally, the algorithm can be modified to handle evolutionary histories of directed graphs. For this, only the recurrence A need be modified. When computing $A'(u, v, f)$, a non-tree edge can be included from u to v, from v to u, both, or neither. Each of these cases modifies the function f in a different way. Specifically:

$$A'(u,v,f) = \begin{cases} 0 + A'(u_L, v, f) + A'(u_R, v, f) \\ 1 + A'(u_L, v, \overleftarrow{f}) + A'(u_R, v, \overleftarrow{f}) \\ 1 + A'(u_L, v, \overrightarrow{f}) + A'(u_R, v, \overrightarrow{f}) \\ 2 + A'(u_L, v, \overleftrightarrow{f}) + A'(u_R, v, \overleftrightarrow{f}), \\ \qquad\qquad\vdots \end{cases}$$

where the vertical ellipsis indicates the symmetric cases involving v_L and v_R, and where $\overrightarrow{f}, \overleftarrow{f}, \overleftrightarrow{f}$ are defined, depending on u and v, as follows:

$$\overrightarrow{f}\ (x,y) = \begin{cases} 1 - f(x,y) & \text{if } x \in \mathrm{ST}(u) \text{ and } y \in \mathrm{ST}(v) \\ f(x,y) & \text{otherwise} \end{cases} \tag{7}$$

$$\overleftrightarrow{f}\ (x,y) = \begin{cases} 1 - f(x,y) & \text{if } x \in \mathrm{ST}(u) \text{ and } y \in \mathrm{ST}(v) \text{ or vice versa} \\ f(x,y) & \text{otherwise,} \end{cases} \tag{8}$$

with $\overleftarrow{f}$ defined analogously to $\overrightarrow{f}$. Here, $\mathrm{ST}(u)$ indicates the set of nodes in the subtree rooted at u.

The heuristic also can be extended to handle different costs for interaction addition and interaction deletion by changing the constants in the recurrences to be functions dependent on f.

4 Results

4.1 Generating Plausible Simulated Histories

We use a *degree-dependent model* (DDM) to simulate an evolutionary path from a putative ancestral network to its extant state. The model simulates node duplication, node deletion, independent interaction gain, and independent interaction loss with given probabilities $\mathrm{P}_{\mathrm{ndup}}$, $\mathrm{P}_{\mathrm{nloss}}$, $\mathrm{P}_{\mathrm{egain}}$ and $\mathrm{P}_{\mathrm{eloss}}$, respectively. The nodes or edges involved in a modification are chosen probabilistically based on their degrees (as in [28]) according to the following expressions:

$$\mathrm{P}(u \mid \text{node duplication}) \propto 1/k_u \qquad \mathrm{P}(u \mid \text{node loss}) \propto 1/k_u \tag{9}$$

$$\mathrm{P}((u,v) \mid \text{interaction gain}) \propto k_u^o \qquad \mathrm{P}((u,v) \mid \text{interaction loss}) \propto 1/k_u^o, \tag{10}$$

where k_u^o is the out-degree of a node u, and k_u is the total degree. At each time step, the distribution of possible modifications to the graph is calculated as $\mathrm{P}(\text{modification}) = \mathrm{P}_{\text{operation}}\mathrm{P}(\text{object} \mid \text{operation})$. Nodes with out-degree of 0 are removed. Varying parameters $\mathrm{P}_{\mathrm{ndup}}$, $\mathrm{P}_{\mathrm{nloss}}$, $\mathrm{P}_{\mathrm{egain}}$ and $\mathrm{P}_{\mathrm{eloss}}$ can produce a wide variety of densities and sizes. We also consider a *degree-independent model*

(DIM) in which the four conditional probabilities in Eqns. (9) and (10) are all equal.

The DDM model is theoretically capable of producing evolutionary trajectories between any two networks while incorporating preferential attachment to the source node and random uniform choice of the target node. Furthermore, choosing a node for duplication or loss in inverse proportion to its degree favors an event in inverse relation to its expected disruption of the network.

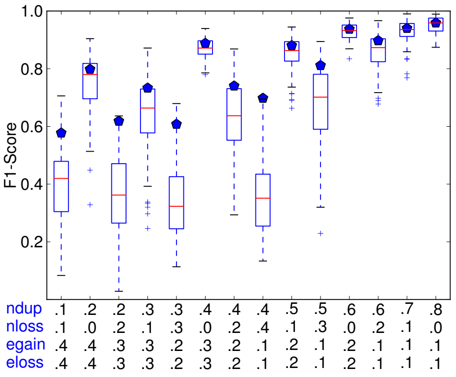
We also consider a model of regulatory network evolution by Foster et al. [12], which is based on gene duplication, with incoming and outgoing interactions kept after duplication as in other models ($\mathrm{P}_{\mathrm{inkeep}}$ and $\mathrm{P}_{\mathrm{outkeep}}$ probabilities respectively). New edges are added with probability $\mathrm{P}_{\mathrm{innovation}}$.

In all of the network evolution models, we started with a random connected seed graph that has 10 nodes and 25 interactions. We evolved it to X by 200 operations after which we introduce a speciation event, and then both G_1 and G_2 evolve from X by an additional 200 operations each. To generate more biologically plausible ancestral graphs, instances were kept only if the ancestral graph X had an in-degree that fit an exponential distribution with parameter between 1.0 and 1.2 or an out-degree that was scale-free with parameter between 1.8 and 2.2.

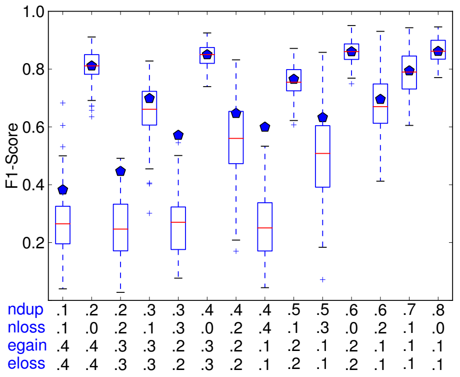
4.2 Reconstructing Histories

Optimality of loop breaking. The greedy procedure to break blocking loops produces histories that are very close to optimal. We generated 1400 networks using the DDM model with the range of parameters on the x-axis of Fig. 3a. In the vast majority of cases (1325 out of 1400), either no loop breaking is required, or the solution discovered after greedily breaking all loops has the same cost as the original solution. In these cases, therefore, the method returned a provably maximally parsimonious set of interaction modification events. In the remaining 75 cases (5.4%), greedily removing blocking loops increased the number of interaction modifications by no more than 10 ($< 2\%$ of the initial number of interaction modification events). Since the initial solution provides a lower bound on the optimal, we can verify that the greedy procedure always found a solution within 2% of the optimal (and perhaps even better). Thus, it seems that in practice, while blocking loops occur, the greedy procedure does a good job of eliminating them without increasing the number of events significantly.

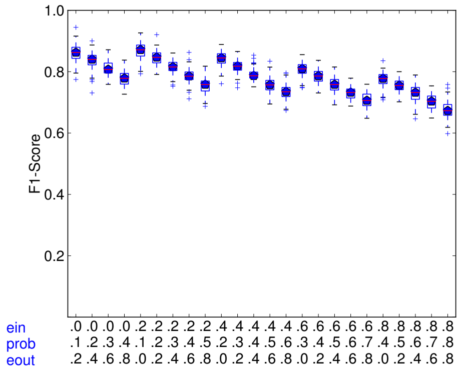
Effect of growth model and its parameters. Modeling the evolutionary dynamics of a regulatory network is still an active topic of research. We therefore experimented with three different network models (Sec. 4.1). Despite their differences, high precision and recall (measured as the F1 score) can be obtained for all of them for many choices of their parameters (Fig. 3a-c). Very good performance can be achieved under the general model presented above whether degree distributions are taken into account (Fig. 3a) or not (Fig. 3b) when selecting nodes and interactions to modify. In these cases, for most parameter choices, precision is close to 1.0, meaning every interaction predicted to be in the ancestor, in fact,



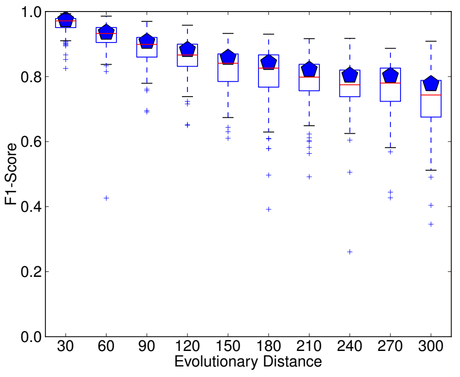
(a) Degree-dependent model (Sec. 4.1)



(b) Degree-independent model



(c) Foster et al. [12] model



(d) Divergence of G_1, G_2 from ancestor

Fig. 3. (a-c) Effect of model parameters on reconstruction accuracy under three different models. "Prob" in (c) is $\mathrm{P}_{\mathrm{innovation}}$. (d) Effect of evolutionary distance (number of network modification operations) on the quality of the ancestral network reconstruction. In both plots, boxes show 1st and 3rd quartile over 100 networks with median indicated by a line. Pentagons show the median if interactions incident to nodes lost in both lineages are not considered.

was. Recall is often lower. The Foster et al. [12] model, with its heavy reliance on duplication events and lack of node loss events, tends to be the simplest under which to reconstruct the ancestral graph (Fig. 3c).

The largest factor leading to poorer performance is lower recall caused by gene losses. If all descendants of a gene are lost in both extant networks, it is not possible to reconstruct interactions incident to it. If these interactions are excluded from the computation of recall, the F1 score often improves dramatically. Median F1 scores excluding these interactions are shown as pentagons in Fig. 3.

Robustness to evolutionary divergence. Naturally, the ability to recover the ancestral network degrades as time passes and the extant networks diverge. However, the degradation is slow (Fig. 3d, using the degree-dependent model with parameters fixed at $P_{ndup} = 0.35$, $P_{nloss} = 0.05$, $P_{egain} = 0.3$, and $P_{eloss} = 0.3$). When the distance is small, we are almost always able to recover the ancestral network well, as illustrated by the high F1-scores and small interquartile ranges in Figure 3d. Even when the distance between the ancestral and extant networks is large (300) compared to the average ancestral network size (55), we obtain an F1-score of 0.72 (0.77 when homology groups lost in both lineages are not considered).

5 Conclusion

We have presented a novel framework for representing network histories involving gene duplications, gene loss, and interaction gain and loss for both directed and undirected graphs. A combinatorial characterization for valid histories was given. We have shown that a fast heuristic can recover optimal histories in a large majority of instances. We further provide evidence that, even with a probabilistic, weighted, generative model of network growth, a parsimony approach can recover accurate histories.

Acknowledgements. This work was partially supported by National Science Foundation grants EF-0849899, IIS-0812111, and CCF-1053918 and National Institutes of Health grant 1R21AI085376 to C.K. G.M. was partially supported by Agriculture and Food Research Initiative Competitive grants 2008-04049 and 2010-15739-01 from the USDA National Institute of Food and Agriculture and NIH grant R01HG002945. The authors thank Geet Duggal and Darya Filippova for helpful discussions.

References

1. Aldana, M., Balleza, E., Kauffman, S., Resendiz, O.: Robustness and evolvability in genetic regulatory networks. J. Theor. Biol. 245(3), 433–448 (2007)
2. Arvestad, L., Berglund, A.C., Sennblad, B.: Bayesian gene/species tree reconciliation and orthology analysis using mcmc. Bioinformatics 19(Suppl. 1), i7–i15 (2003)
3. Borenstein, E., Feldman, M.W.: Topological signatures of species interactions in metabolic networks. J. Comput. Biol. 16(2), 191–200 (2009)
4. Borenstein, E., Kupiec, M., Feldman, M.W., Ruppin, E.: Large-scale reconstruction and phylogenetic analysis of metabolic environments. Proc. Natl. Acad. Sci. USA 105(38), 14482–14487 (2008)
5. Chen, K., Durand, D., Farach-Colton, M.: NOTUNG: A program for dating gene duplications and optimizing gene family trees. Journal of Computational Biology 7(3-4), 429–447 (2000)
6. Chung, F., Lu, L., Dewey, T.G., Galas, D.J.: Duplication models for biological networks. J. Comp. Biol. 10(5), 677–687 (2003)
7. Durand, D., Halldrsson, B.V., Vernot, B.: A hybrid micromacroevolutionary approach to gene tree reconstruction. J. Comp. Biol. 13(2), 320–335 (2006)

8. Dutkowski, J., Tiuryn, J.: Identification of functional modules from conserved ancestral proteinprotein interactions. Bioinformatics 23(13), i149–i158 (2007)
9. Erten, S., Li, X., Bebek, G., Li, J., Koyuturk, M.: Phylogenetic analysis of modularity in protein interaction networks. BMC Bioinformatics 10, 333 (2009)
10. Espinosa-Soto, C., Martin, O.C., Wagner, A.: Phenotypic robustness can increase phenotypic variability after nongenetic perturbations in gene regulatory circuits. J. Evol. Biol. 24(6), 1284–1297 (2011)
11. Flannick, J., Novak, A., Srinivasan, B.S., McAdams, H.H., Batzoglou, S.: Graemlin: general and robust alignment of multiple large interaction networks. Genome Res. 16(9), 1169–1181 (2006)
12. Foster, D.V., Kauffman, S.A., Socolar, J.E.S.: Network growth models and genetic regulatory networks. Phys. Rev. E 73(3), 031912 (2006)
13. Gibson, T.A., Goldberg, D.S.: Reverse engineering the evolution of protein interaction networks. In: Pac. Symp. Biocomput., pp. 190–202 (2009)
14. Ispolatov, I., Krapivsky, P.L., Yuryev, A.: Duplication-divergence model of protein interaction network. Phys. Rev. E 71(6 Pt 1), 061911 (2005)
15. Kreimer, A., Borenstein, E., Gophna, U., Ruppin, E.: The evolution of modularity in bacterial metabolic networks. Proc. Natl. Acad. Sci. USA 105(19), 6976–6981 (2008)
16. Kuchaiev, O., Milenkovic, T., Memisevic, V., Hayes, W., Przulj, N.: Topological network alignment uncovers biological function and phylogeny. J. R. Soc. Interface 7(50), 1341–1354 (2010)
17. Levy, E.D., Pereira-Leal, J.B.: Evolution and dynamics of protein interactions and networks. Curr. Opin. Struct. Biol. 18(3), 349–357 (2008)
18. Middendorf, M., Ziv, E., Wiggins, C.H.: Inferring network mechanisms: the *Drosophila melanogaster* protein interaction network. Proc. Natl. Acad. Sci. USA 102(9), 3192–3197 (2005)
19. Mirkin, B.G., Fenner, T.I., Galperin, M.Y., Koonin, E.V.: Algorithms for computing parsimonious evolutionary scenarios for genome evolution, the last universal common ancestor and dominance of horizontal gene transfer in the evolution of prokaryotes. BMC Evol. Biol. 3, 2 (2003)
20. Mithani, A., Preston, G., Hein, J.: A stochastic model for the evolution of metabolic networks with neighbor dependence. Bioinformatics 25(12), 1528–1535 (2009)
21. Navlakha, S., Kingsford, C.: Network archaeology: Uncovering ancient networks from present-day interactions. PLoS Comput. Biol. 7(4), e1001119 (2011)
22. Pachter, L.: An introduction to reconstructing ancestral genomes. In: Proc. Symp. in Applied Mathematics, vol. 64, pp. 1–20 (2007)
23. Pastor-Satorras, R., Smith, E., Sole, R.: Evolving protein interaction networks from gene duplication. J. Theor. Biol. 222, 199–210 (2003)
24. Pereira-Leal, J.B., Levy, E.D., Kamp, C., Teichmann, S.A.: Evolution of protein complexes by duplication of homomeric interactions. Genome Biol. 8(4), R51 (2007)
25. Pinney, J.W., Amoutzias, G.D., Rattray, M., Robertson, D.L.: Reconstruction of ancestral protein interaction networks for the bZIP transcription factors. Proc. Natl. Acad. Sci. USA 104(51), 20449–20453 (2007)
26. Raman, K., Wagner, A.: Evolvability and robustness in a complex signalling circuit. Mol. Biosyst. 7(4), 1081–1092 (2011)
27. Singh, R., Xu, J., Berger, B.: Pairwise global alignment of protein interaction networks by matching neighborhood topology. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 16–31. Springer, Heidelberg (2007)

28. Stewart, A.J., Seymour, R.M., Pomiankowski, A.: Degree dependence in rates of transcription factor evolution explains the unusual structure of transcription networks. Proc. Biol. Sci. 276(1666), 2493–2501 (2009)
29. Teichmann, S.A., Babu, M.M.: Gene regulatory network growth by duplication. Nat. Genetics 36(5), 492–496 (2004)
30. Zhang, X., Moret, B.: Refining transcriptional regulatory networks using network evolutionary models and gene histories. Alg. Mol. Biol. 5(1), 1 (2010)
31. Zhang, X., Moret, B.M.: Boosting the performance of inference algorithms for transcriptional regulatory networks using a phylogenetic approach. In: Crandall, K.A., Lagergren, J. (eds.) WABI 2008. LNCS (LNBI), vol. 5251, pp. 245–258. Springer, Heidelberg (2008)

A Combinatorial Framework for Designing (Pseudoknotted) RNA Algorithms

Yann Ponty[1,⋆] and Cédric Saule[2,3]

[1] LIX, École Polytechnique/CNRS/INRIA AMIB, France
yann.ponty@lix.polytechnique.fr
[2] LRI, Université Paris-Sud/XI/INRIA AMIB, France
[3] Institute for Research in Immunology and Cancer, Montreal, Quebec, Canada
saule@lri.fr

Abstract. We extend an hypergraph representation, introduced by Finkelstein and Roytberg, to unify dynamic programming algorithms in the context of RNA folding with pseudoknots. Classic applications of RNA dynamic programming (Energy minimization, partition function, base-pair probabilities...) are reformulated within this framework, giving rise to very simple algorithms. This reformulation allows one to conceptually detach the conformation space/energy model – captured by the hypergraph model – from the specific application, assuming unambiguity of the decomposition. To ensure the latter property, we propose a new combinatorial methodology based on generating functions. We extend the set of generic applications by proposing an exact algorithm for extracting generalized moments in weighted distribution, generalizing a prior contribution by Miklos and al. Finally, we illustrate our full-fledged programme on three exemplary conformation spaces (secondary structures, Akutsu's simple type pseudoknots and kissing hairpins). This readily gives sets of algorithms that are either novel or have complexity comparable to classic implementations for minimization and Boltzmann ensemble applications of dynamic programming.

Keywords: RNA folding, Pseudoknots, Boltzmann Ensemble, Hypergraphs, Dynamic Programming.

1 Introduction

Motivation. Over the past decades biology as a field has become increasingly aware of the importance and diversity of roles played by ribonucleic acids (RNA). In addition to playing house-keeping parts, as initially contemplated by the proteo-centric view of cellular processes, RNA is now accepted as a major player of gene regulation mechanisms. For instance silencing activity (miRNAs, siRNAs) or multi-stable cis-regulatory elements (riboswitches) are currently the subject of many research. Furthermore a recent genome-wide experiment has revealed that a large portion of the human genome was subject to transcription

⋆ To whom correspondence should be addressed.

into RNA. While it is unlikely for all these transcripts to be functional as RNAs, novel classes and roles are currently under investigation. Most of the functional roles played by RNA require the RNA to adopt a specific structure to make an interaction possible, hide/exhibit an active site or allow for a catalytic action (Ribozymes). Being able to understand and simulate how RNA folds is therefore a crucial step toward understanding its function.

Ab initio secondary structure prediction. Initial algorithmic methods for the ab-initio prediction of RNA folding considered a coarse-grain conformation space, the secondary structure, where each conformation is defined as a non-crossing subset of admissible base-pairs. This led Nussinov and Jacobson ([39]) to design a $\Theta(n^3)$ dynamic-programming (DP) algorithm for the base-pair maximization problem. Building on a nearest neighbor free-energy model proposed by Tinoco *et al* ([51]) and extended by the Turner group, Zuker and Stiegler ([56]) created MFold, a $\Theta(n^3)$ algorithm for minimizing the free-energy (MFE folding), later shown to predict correctly ∼73% of base-pairs on a benchmark of RNAs of length < 700 nucleotides ([34]). An independent implementation of the algorithm is proposed within the popular ViennaRNA package maintained by Hofacker ([22]). Probabilistic alternatives (SFold ([11]), ContraFold ([14]) and CentroidFold ([20])) have also recently been proposed with substantial improvement, relying on a dynamic programming scheme similar to that of MFold to traverse the conformation space in polynomial time coupled with some post-processing steps.

Ensemble approaches. Since the seminal work of McCaskill ([35]), the concept of Boltzmann equilibrium has been used to embrace the diversity of folding accessible to an RNA sequence. He showed that the partition function of an RNA – a weighted sum over the set of all compatible structures – could be computed through a simple transposition of the DP scheme used for MFE folding. Coupled with a variant of the inside/outside algorithm, this led to an exact computation of base-pairs probabilities in the Boltzmann-weighted ensemble. This opened the door for more robust predictions, e.g. for RNAs whose MFE folding is an outlier. This intuition was later validated by Mathews ([33]) who showed that the Boltzmann probability correlated well with the actual presence of base-pairs in experimentally-determined structures. Ding *et al* ([11]) pushed this paradigm shift a step further by clustering sets of structures sampled within the Boltzmann distribution and computing a consensus, improving on the positive-predictive-value (PPV) of existing algorithms. This ensemble view naturally spread toward other applications of DP in Bioinformatics (sequence alignement ([36]), simultaneous alignment and folding ([21]), 3D structural alignement ([15])), and is increasingly becoming a part of the *algorithmic toolbox* of bioinformaticians.

Pseudoknotted conformations. Although substantially successful in their task, secondary structure prediction algorithms were intrinsically limited in by their inability to explore conformations featuring crossing base-pairs. Such motifs, called pseudoknots, were initially excluded from the conformation space based on the rationale that their participation to the free-energy would remain limited. Furthermore, the adjunction of all possible pseudoknots was shown to turn MFE folding into an NP-complete problem even in a simple nearest-neighbor model ([1]; [30]). However such conformations do naturally occur, and can be essential to functional mechanisms such as -1-frameshift recoding events ([4]) or the formation of tertiary motifs ([40]). Therefore many exact DP approaches ([45]; [30]; [13]; [42]; [6]; [7]; [8]; [7]; [23]; [50]; [44]) have been proposed over the years to extract the MFE structure within restricted – polynomially solvable – classes of pseudoknots. However most of these approaches (with the notable exceptions of ([13]; [6]; [44])) were based on ambiguous DP schemes, leading them to consider certain structures multiple times. While such an unambiguity would not be worrisome in the context of energy minimization, it prevents a direct transposition of these algorithms to ensemble applications (partition function, base-pair probabilities) by heavily biasing – for no biologically valid reason – derived estimates.

Unambiguous decompositions. This lack of focus on unambiguity in the design of RNA (pseudoknotted) DP algorithms can be explained by two main reasons. Firstly certain conformation spaces may not admit unambiguous schemes. Indeed it has been shown by Condon *et al* ([9]) that many PK conformational spaces can be modeled as a formal language, while Flajolet ([18]) had shown, using a combinatorial argument, that certain simple context-free languages are inherently ambiguous, i.e. not generated by any unambiguous context-free grammar. A second explanation is more historical: DP algorithms designers were initially focused on optimization problems, and considered the DP equation, not the decomposition of the search space, as the central object of their contributions. Indeed in the optimization perspective, it is not mandatory for the conformation space to be completely (e.g. sparsification) or unambiguously (e.g. multiply occurring best structure) generated. As decompositions grow more and more complex to capture more complex energy models and topological limitations, these two key properties are becoming increasingly hard to ascertain at the level of DP equations. Consequently there is a need for more rational framework to facilitate the design of conformational spaces.

Combinatorial dynamic programming. Over the last century, enumerative combinatorics as a field has been focusing on providing elegant decompositions for all sorts of objects. Our proposal is to adopt a similar discipline in the design of DP decompositions, the only task worthy of human attention to our opinion, and will eventually lead to an automated procedure for the actual production of codes/algorithms. To that purpose we chose to build on and revisit an hypergraph analogy proposed by Finkelstein *et al* ([16]) as a unifying framework for RNA folding and other applications of DP in Bioinformatics, which we generalize into combinatorial classes amenable to analysis using generating functions.

Related work. The two main frameworks offering abstracts view over Dynamic Programming are Lefebvre's multi-tape attributed grammars ([26]) and Giegerich's Algebraic Dynamic Programming (ADP) ([19]), respectively building

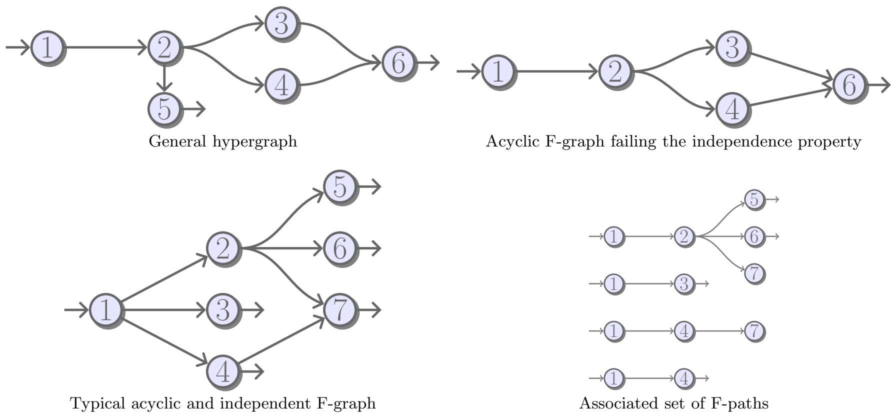

1
2
3
4
5
6
General hypergraph
Acyclic F-graph failing the independence property
Typical acyclic and independent F-graph
7
Associated set of F-paths

Fig. 1. Illustration of F-Graphs, F-Paths and Independence property. Straight lines indicate classic arcs, and bent lines indicate hyperarcs.

on multitape-attributed grammars and context-free grammars. Although very elegant and mature in their implementations, they suffer from limitations in expressivity that are intrinsic to their underlying formalisms. For instance, ADP has to resort to an explicit manipulation of indices in order to achieve competitive complexities for canonical pseudoknots ([42]), while Lefebvre's multi-tape grammars ([27]) require increased complexity to capture pseudoknots. Another formal description of pseudoknotted search spaces is M. Möhl's *split-types* ([37]), which focuses on how non-contiguous portions are combined, providing a very compact description for pseudoknotted conformation spaces. Compared to these abstract representations, the hypergraph formalism achieves a greater expressivity by: i) Implementing an unordered product; ii) Allowing explicit manipulation of indices; iii) Allowing additional information to be stored within nodes (Remember that context-free grammars allow for a finite number of non-terminals). For instance, polynomial hypergraphs could be proposed for counting homogeneous alignments ([25]) whereas these objects cannot be generated by any context-free grammar ([5]) and will not be expressed strictly within the alternative frameworks. This improved expressivity comes at a price since the manual manipulation of indices is error-prone, as pointed accurately by Giegerich et al, so one may want to think of our proposal as more of a byte code, possibly produced from a higher-level source code (ADP, split-types. . .).

Outline. In Section 2, we briefly remind some basic definitions related to forward directed hypergraphs. In Section 3, we remind and propose dynamic programming algorithms for generic problems on F-graphs. Then in Section 4, we illustrate our programme by proposing and proving unambiguous decompositions for three space of conformations: Classic secondary structures in the Turner energy model ([32]), (weighted) base-pair maximisation version of Akutsu's simple-type pseudoknots ([1]) and fully-recursive kissing hairpins (Unambiguous restriction of Chen *et al* ([8])). We also describe a simplified proof strategy based on generating functions to prove the correctness of a given decomposition. Section 5 enriches the scope of applications of our framework by proposing a general algorithm for extracting the moments of additive features (free-energy, base-pairs, helices…) in a weighted distribution (generalizing a previous contribution by Miklos *et al* ([38])). Finally Section 6 concludes with some remarks and possible extensions and improvements.

2 Notations and Key Notions

Let us first remind that a directed hypergraph generalizes the notion of directed graph by allowing any number of vertices as origin(**tail**) and destination (**head**) for each (hyper)-arcs. We will be focusing here on Forward-Hypergraphs, or **F-graphs**, which restrict the tail of their arcs to a single vertex.

Formally, let V be a set of vertices, an **F-arc** $e = (t(e) \to \mathbf{h}(e)) \in V \times \mathcal{P}(V)$, connects a single tail vertex $t(e) \in V$ to an ordered list of vertices $\mathbf{h}(e) \subseteq V$. An **F-graph** $\mathcal{H} = (V, E)$ is characterized by a set of vertices V and a set of F-arcs E. Denote by $\mathbf{c}_n$ the children of a node in a tree, then an **F-path** of $\mathcal{H} = (V, E)$ is a tree $\mathcal{T} = (V' \subseteq V, E')$ such that, for any node $n \in V'$, $(v_n \to \mathbf{c}_n) \in E$. For the sake of simplicity, we may omit the implicit V' and identify an F-path to its set of edges E'.

An **F-derivation** from a vertex $s \in V$ can be recursively defined as either $\langle s, \varnothing\rangle$ if $(s \to \varnothing) \in E$, or $\langle s, D_1 \ldots D_{|\mathbf{t}|}\rangle$ if $(s \to \mathbf{t}) \in E$, $\mathbf{t} = \{t_1, t_2, \ldots, t_{|t|}\}$, and each D_i is an F-derivation starting from t_i. An F-graph is **acyclic** if and only if any vertex $s \in V$ is present only once (as a root) in any derivations starting from s. Moreover it is **independent** if and only if any vertex $s \in V$ is reached at most once in any derivation, regardless of its root.

A **weighted F-graph** is a triplet (V, E, π) such that (V, E) is an F-graph and $\pi : E \to \mathbb{R}^+$ is a weight function that associates a weight to each F-arc. Finally, an **oriented F-graph** is a quadruplet (v_0, V, E, π) such that (V, E, π) is a weighted independent F-graph, and $v_0 \in V$ is a distinguished initial vertex.

Remark 1: Notice that our definition of F-arcs and F-paths implicitly defines **terminal vertices**, since any leaf l in a F-path has no child and our definition of F-paths therefore requires $l \to \varnothing$ to be an F-arc of $\mathcal{H}$.

Remark 2: Under the independence property, the derivations starting from any node $s \in V$ are trees, and are therefore in bijection with F-paths originating from the same vertex.

3 Generic Problems and Algorithms for F-Paths in F-Graphs

In the following, terminal cases will very seldom appear explicitly, but will rather be captured by the limit cases of products $\prod_{u\in\varnothing} f(u) = 1$ and sums $\sum_{u\in\varnothing} f(u) = 0$, $k \in \mathbb{R}$.

Generating and counting F-paths in oriented F-graphs. Let $\mathcal{H} = (v_0, V, E, \pi)$ be an oriented F-graph, we address the problem of generating the set $\mathcal{P}_{v_0}$ of F-paths obtained starting from v_0.

From the tree-like definition of F-paths and our remark on terminal vertices, we know that any F-path starting from a vertex s can either be a leaf, provided that there exists an F-arc $s \to \varnothing$, or an internal node. In the latter case, any F-paths is composed of auxiliary paths, generated from the vertices in the head of some F-edge having s as tail. Remark that our definition of F-paths requires each vertex from V to appear at most once in any F-path, a fact that is ensured here by the acyclicity of $\mathcal{H}$. Therefore we can recursively define the set of $\mathcal{P}_s$ of F-paths starting from a root node s as

$$\mathcal{P}_s = \left\{ \begin{array}{cc} \{(s,\varnothing)\} & \text{If } (s,\varnothing) \in E \\ \varnothing & \text{Otherwise} \end{array} \right\} \cup \bigcup_{(s\to\mathbf{t})\in E} \left(\{s\} \times \prod_{u\in\mathbf{t}} \mathcal{P}_u \right), \quad \forall s \in V. \quad (1)$$

Since E is a set, the candidate heads for a given tail s are distinct and the unions in the above equations are disjoint. Furthermore, the products are Cartesian, so we can directly transpose the recurrence above over the cardinalities $n_s = |\mathcal{P}_s|$ and obtain

$$n_s = \sum_{(s\to\mathbf{t})\in E} \prod_{u\in\mathbf{t}} n_u, \quad \forall s \in V. \quad (2)$$

This immediately yields a $\Theta(|V| + |E| + \sum_{e\in E} |\mathbf{h}(e)|)/\Theta(|V|)$ time/memory dynamic programming algorithm for counting F-paths.

Minimal score F-path. Let us consider an **additive scoring scheme** based on weights, and accordingly define the **score** of an F-path p to be $\alpha(p) = \sum_{e\in E} \pi(e)$. We address here the problem of finding an F-path p_0 having minimal score or more formally some $p_0 \in \mathcal{P}_{v_0}$ such that $\forall p \in \mathcal{P}_{v_0},\ p \neq p_0 \Rightarrow \alpha(p) \geq \alpha(p_0)$. From the independence of siblings and the strict additivity of the score, we know that the path minimization problem has optimal substructure, i. e. any optimal solution is composed of optimal solutions for its subproblems. Consequently, the **minimal score** m_s of a path starting from a root node $s \in V$ is given by

$$m_s = \min_{e=(s\to\mathbf{t})\in E} \left(\pi(e) + \sum_{u\in\mathbf{t}} m_u \right), \quad \forall s \in V. \quad (3)$$

A classic backtrack procedure can then be used to reconstruct the F-path instance $p_s^{\min}$ starting from $s \in V$ and having minimal score. Alternatively, the previous recurrence can be modified as follows

$$p_s^{\min} = \mathop{\arg\min}_{\substack{p' = \bigcup_{s'\in\mathbf{t}} p_{s'}^{\min} \\ \text{s.t. } (s\to\mathbf{t})\in E}} \alpha\left(\{(s \to \mathbf{t})\} \cup p'\right), \quad \forall s \in V, \quad (4)$$

giving a $\Theta(|V| + |E| + \sum_{e\in E} |\mathbf{h}(e)|)/\Theta(|V|)$ time/memory DP algorithm for the minimal weighted F-path.

Weighted count and weighted random generation. Let us **extend multiplicatively on paths** our weight function, defining the **weight of any F-path** p to be $\pi(p) = \prod_{e\in p} \pi(e)$. Then a small modification of Equation 2 gives a recurrence for computing the cumulated weight, or **weighted count** w_s of F-paths starting from a given vertex s:

$$w_s = \sum_{p'\in\mathcal{P}_s} \pi(p') = \sum_{e=(s\to\mathbf{h}(e))\in E} \pi(e) \cdot \prod_{s'\in\mathbf{h}(e)} w_{s'}, \quad \forall s \in V \tag{5}$$

Provided that the weights are positive, this defines a **weighted probability distribution** over F-paths, which assigns to each path $p \in \mathcal{P}_{v_0}$ a probability

$$\mathbb{P}(p \mid \pi) = \frac{\pi(p)}{\sum_{p'\in\mathcal{P}_{v_0}} \pi(p')} \equiv \frac{\pi(p)}{w_{v_0}}. \tag{6}$$

From the precomputed values w_s, one can perform a **weighted random generation** to draw at random a set of k F-paths from v_0 according to a weighted distribution. Starting from any vertex s, the algorithm chooses at each step an F-arc $e = (s \to \mathbf{h}(e))$ with probability

$$p_{s,e} = \frac{\pi(e)\cdot\prod_{s'\in\mathbf{h}(e)} w_{s'}}{w_s},$$

and proceeds to the recursive generation of auxiliary paths from each vertex in $\mathbf{h}(e)$. A simple induction argument shows that any F-path is then generated with respect to the probability distribution of Equation 6. The weighted count recurrence is computed by a $\Theta(|V| + |E| + \sum_{e\in E} |\mathbf{h}(e)|)/\Theta(|V|)$ time/memory algorithm, and each path p is generated in $\Theta(|p|+\sum_{e\in p} |\mathbf{h}(e)|)/\Theta(|p|)$ time/memory.

Remark 3: This worst-case complexity can be improved using additional information on the structure of the F-graph. For instance, when both the height and maximal degree of a vertex are bounded by some constant n, Boustrophedon search ([17]; [41]) can be used to decrease the worst-case complexity of each generation from $\times(n^2)$ to $\mathcal{O}(n \log n)$.

Arc traversal probabilities. Using the same probability distribution, a natural problem is to compute the probability p_e of an F-arc $e \in E$ being in a random F-path. To that purpose one can use the classic *inside/outside* algorithm, which can be rephrased as an F-graphs traversal.

Let us first point out that the probability p_e is related to the cumulated weight of all F-paths featuring an edge $e = (t(e) \to \mathbf{h}(e))$ through

$$p_e = \frac{\sum_{\substack{p\in\mathcal{P}_{v_0}\\ \text{s.t. } e\in p}} \pi(p)}{\sum_{p'\in\mathcal{P}_{v_0}} \pi(p')} \equiv \frac{\sum_{\substack{p\in\mathcal{P}_{v_0}\\ \text{s.t. } e\in p}} \pi(p)}{w_{v_0}}. \tag{7}$$

From the independence of $\mathcal{H}$, we know that each vertex appears at most once in any given F-path, and consequently any F-path traversing e can therefore be **unambiguously** decomposed into: i) An **e-outside tree**, i.e. a derivation from v_0 whose leaves are either terminal or $t(e)$, and which features exactly one occurrence of $t(e)$; ii) A **support edge** $e = (t(e) \to \mathbf{h}(e))$; iii) An **e-inside tree**, i.e. a set of F-paths issued from $\mathbf{h}(e)$.

The unambiguity of the decomposition, along with the independence of i) and iii), translates into

$$\sum_{\substack{p\in\mathcal{P}_{v_0}\\ \text{s.t. } e\in p}} \pi(p) = b_{t(e)} \cdot \pi(e) \cdot \prod_{s'\in\mathbf{h}(e)} w_{s'} \tag{8}$$

where b_s is the cumulated weight of all outside trees leaving $s \in V$ underived. Finally it can be shown that the cumulated weight b_s over all e-outside trees obey the following simple recurrence

$$b_s = \mathbf{1}_{s=q_0} + \sum_{\substack{e'\in E\\ \text{s. t. } s\in\mathbf{h}(e')}} \pi(e') \cdot b_{t(e')} \cdot \prod_{\substack{s'\in\mathbf{h}(e')\\ s'\neq s}} w_{s'}, \quad \forall s \in V \tag{9}$$

which can computed in $O(|V| + |E| + \sum_{e\in E} |\mathbf{h}(e)|^2)/\Theta(|V|)$ time/memory. The probability of traversing p_e in a random F-path can finally be computed through the formula

$$p_e = \frac{b_{t(e)} \cdot \prod_{s'\in\mathbf{h}(e)} w_{s'}}{w_{v_0}}, \quad \forall e \in E. \tag{10}$$

4 F-Graphs Reformulation of (Pseudoknotted) RNA Conformation Spaces

From the previous section, we know that very simple algorithms exist for weighted optimization and enumeration problems over the F-paths of an F-graph. Let us now consider MFE folding-related problems over an arbitrary **conformation space** D for a sequence ω, under an energy model $E : D \to \mathbb{R}$ and assume that there exists: **C1.** An F-graph $\mathcal{H}$ whose F-paths $\mathcal{P}$ are in bijection with the conformation space D; **C2.** A weight function π such that the (additive) score of any F-path coincides with the energy of its corresponding conformation.

Under such conditions, it can be remarked that the **minimal score** algorithm (Equation 3) exactly computes the **Minimal Free-Energy** $MFE = \min_{s\in D} E_s$. Furthermore, the **Weighted Count** (Equation 5), applied to a weight function $\pi'(e) = e^{-\pi(e)/RT}$, computes the **Partition Function** $\mathcal{Z} = \sum_{s\in D} e^{-E_s/RT}$. Other quantities of interest for RNA folding can also be derived, as summarized in Tables 1 and 2.

4.1 Foreword: Shortening Correctness Proofs through Generating Functions

Our main challenge is to find an hypergraph/weight such that the energy function can be expressed in an additive fashion. Focusing first on Condition **C1**, one remarks that finding a function $\psi : \mathcal{P} \to D$ which maps F-Paths to elements of the conformation space is not challenging, as it essentially amounts to figuring out which derivation creates which base-pairs. Condition **C1** is then traditionally broken into two parts: an **unambiguity** condition which requires distinct elements in $\mathcal{P}$ to give rise to distinct elements within D, i.e. ψ should be injective; a **completeness** condition which requires each element in S to have at least one pre-image, i.e. ψ should be surjective.

Since these notions are intimately related to the semantics associated with the F-paths, they cannot be tackled in an automated way at the hypergraph level[1]. Therefore correctness proofs will usually require user-assigned semantics coupled with custom arguments, a task that may become challenging and/or tedious for complex decompositions. In order to simplify the validation and therefore the design of new conformation spaces, we propose a simplified proof technique based on generating functions.

Indeed, instead of specializing the hypergraph for each and every input sequence, one can delegate to the weight function the responsibility of weeding out conformations, e.g. by assigning them $+\infty$ energetic contributions within MFE folding. Therefore each class of conformations can be seen as a family of conformation space $\{D_n\}_{n\geq 0}$ (secondary structures, simple type pseudoknots...), to which one associates a family of hypergraphs $\{\mathcal{H}_n\}_{n\geq 0}$, a **decomposition**, both indexed by the length n of the sequence.

Let us remind that generating functions are formal power series that can be used to store various information. For instance the counting generating function for the conformation space family $\mathcal{D}$ can be defined as $S_{\mathcal{D}}(z) = \sum_{n\geq 0} |D_n| \cdot z^n$ where z is a formal complex variable devoid of intuitive meaning. Furthermore let $\mathcal{P}_n$ be the set of F-Paths associated with $\mathcal{H}_n$, then the counting generating function of the decomposition can be defined as $S_{\mathcal{H}}(z) = \sum_{n\geq 0} |\mathcal{P}_n| \cdot z^n$. Then the formal identity $S_{\mathcal{D}}(z) = S_{\mathcal{H}}(z)$ implies that $|D_n| = |\mathcal{P}_n|, \forall n \geq 0$. It follows from basic set theory that unambiguity/injectivity (resp. completeness/surjectivity) of ψ, in addition to the identity of generating functions, is in itself sufficient to prove the bijectivity of ψ. Since reference generating functions are now available for many conformation space families ([47]), this practically halves the burden of designing a proof.

4.2 RNA Secondary Structures

Let us first illustrate our approach on RNA secondary structures, for which `Unafold` ([32]) – the successor of `MFold` ([56]) – offers an unambiguous scheme. Compared to the original decomposition presented in Markham's thesis ([31]), the one described in Figure 2 is simplified to ignore dangles.

Proving unambiguity

- Let us remark that both Q^5 and Q^1 either leave their last base j unpaired (Left), or pairs it to i (Right). Furthermore these two cases are mutually exclusive. Finally Q^1 generates exactly one helix.

[1] Algebraic Dynamic Programming partially addresses this issue, and the interested reader is referred to an early contribution by Reeder *et al* ([43]).

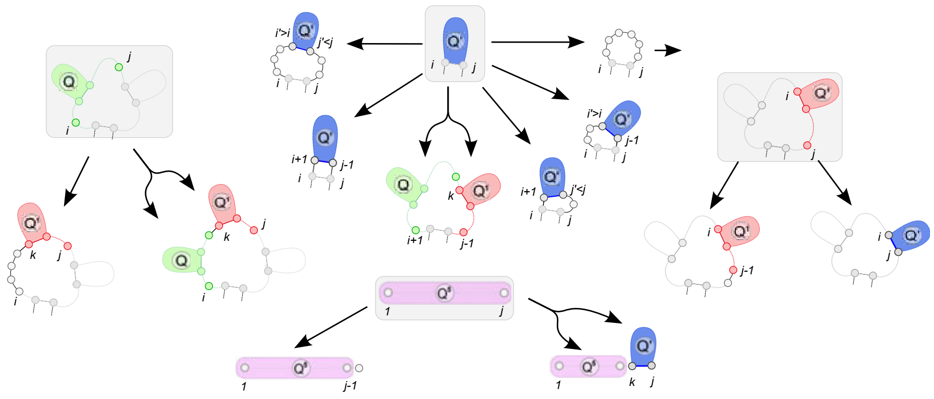


Fig. 2. Simplification of the `Unafold` ([32]) decomposition of the secondary structures space. Framed states indicate origins of (hyper)arcs.

– Q always makes at least one call to Q^1 and therefore creates at least one helix. Therefore, it either creates exactly one helix (Left case) or more (Right case), and these two cases are mutually exclusive.
– Q' distinguishes different types of loops. Let m_5, m_3 be the numbers of unpaired bases on the $5'$ strand, $3'$ strand, and h be the number of helices starting from case Q', one can label each of the cases and observe that they are mutually non-overlapping. Namely from left to right, we get the following (m_5, m_3, h) triplets: Interior loop $(>0, >0, 1)$, stacking pair $(0,0,1)$, multiloop $(\geq 0, \geq 0, >1)$, bulges $5'$ $(>0,0,1)$ and $3'$ $(0,>0,1)$, and hairpin loop $(>0,>0,0)$.

Deriving completeness. From previous work by Waterman ([54]), we know that the generating function of secondary structures with at least one unpaired base between paired bases ($\theta = 1$) is

$$S(z) = \frac{1 - z + z^2 - \sqrt{1 - 2z - z^2 - 2z^3 + z^4}}{2z^2}. \tag{11}$$

Following the general principle of the so-called DSV methodology (See Lorenz *et al* ([29]) for a presentation in a similar context), the `Unafold` decomposition can be translated into a system of algebraic equations. Namely, one simply replaces any occurrence of k unpaired base with z^k, each basepair with z^2, and any vertex with its associated generating function. Let $Q^5(z)$, $Q(z)$, $Q'(z)$ and $Q^1(z)$ be the generating functions counting the F-paths generated from Q^5, Q, Q' and Q^1 respectively:

$$Q^5(z) = Q^5(z)\cdot z + Q^5(z)\cdot Q'(z) \qquad Q(z) = \mathrm{Seq}(z)\cdot Q^1(z) + Q(z)\cdot Q^1(z) \qquad Q^1(z) = z\cdot Q^1(z) + Q'(z)$$

$$\begin{aligned} Q'(z) =& z^2\cdot \mathrm{Seq}^+(z)\cdot Q'(z)\cdot \mathrm{Seq}^+(z) + z^2\cdot Q'(z) + z^2\cdot Q(z)\cdot Q'(z)\\ &+ z^2\cdot Q'(z)\cdot \mathrm{Seq}^+(z) + z^2\cdot \mathrm{Seq}^+(z)\cdot Q^1(z) + \mathrm{Seq}^+(z)\end{aligned}$$

$$\mathrm{Seq}^+(z) = z\cdot \mathrm{Seq}(z) \qquad \mathrm{Seq}(z) = z\cdot \mathrm{Seq}(z) + 1.$$

Table 1. Reformulations of secondary structure applications as F-graphs problems and associated complexities

Application	Algorithm	Weight fun.	Time	Memory	Ref.
A – Energy minimization	Minimal weight	$\pi_\mathcal{T}$	$O(n^3)$	$O(n^2)$	(56)
B – Partition function	Weighted count	$e^{\frac{-\pi_\mathcal{T}}{RT}}$	$O(n^3)$	$O(n^2)$	(35)
C – Base-pairing probabilities	Arc-traversal prob.	$e^{\frac{-\pi_\mathcal{T}}{RT}}$	$O(n^3)$	$O(n^2)$	(35)
D – Statistical sampling (k-samples)	Weighted random gen.	$e^{\frac{-\pi_\mathcal{T}}{RT}}$	$O(n^3 + k \cdot n \log n)$	$O(n^2)$	(12; 41)
E – Moments of energy (Mean, Var.)	Moments extraction	$e^{\frac{-\pi_\mathcal{T}}{RT}}$	$O(n^3)$	$O(n^2)$	(38)
F – m-th moment of additive features	Moments extraction	$e^{\frac{-\pi_\mathcal{T}}{RT}}$	$O(m^3 \cdot n^3)$	$O(m \cdot n^2)$	–
G – Correlations of additive features	Moments extraction	$e^{\frac{-\pi_\mathcal{T}}{RT}}$	$O(n^3)$	$O(n^2)$	–

Solving the system yields $Q^5(z) = S(z)$ which, in conjunction with the unambiguity of the decomposition, proves its completeness.

Applicability of generic algorithms. Let us show that $\mathcal{H}$ fulfills the prerequisites of our algorithms. First it is easily verified that $\mathcal{H}$ is an F-graph. Associating a region $[i, j]$ (resp. $[1, j]$) with each vertex $q^1_{i,j}$, $q_{i,j}$ and $q'_{i,j}$ (resp. q^5_j), one easily verifies that for any F-arc $e \in E$ the width of any region in the head $\mathbf{h}(e)$ is strictly smaller than that of the tail $t(e)$, and the **acyclicity** of $\mathcal{H}$ directly follows. Furthermore, any two vertices in the head $\mathbf{h}(e)$ have non-overlapping associated regions. Consequently $\mathcal{H}$ is **independent**, and a direct application of our generic algorithms gives a set of algorithms summarized in Table 1. This gives a family of efficient $O(n^3)$ algorithms for assessing RNA secondary structure properties at the Boltzmann equilibrium.

Remark 4: In interior loops, the set of F-arcs generated for the Q' case has apparent cardinality in $O(n^4)$. This can be brought back to $O(n^3)$ by enforcing constraints on the energy function. Traditionally, the accepted practice is to bound the interior loop *size* $(j' - j) + (i' - i)$ from above by a predefined constant $K \approx 30$. Exhaustive $O(n^3)$ decompositions can also be proposed (Figure 3) by decomposing the internal loop into additively-contributing regions. A first option may generate independently the left and right unpaired regions (Figure 3, Left), while an alternative may decompose internal loops into a symmetric loop followed by a fully asymmetric one (Figure 3, Right).

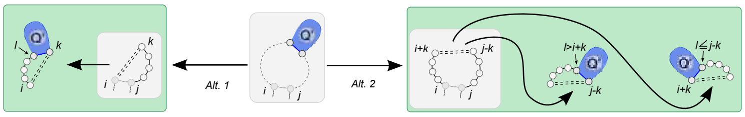


Fig. 3. Alternative exhaustive strategies for interior loops

4.3 Simple-Type Pseudoknots

In his seminal work, Akutsu (1) focused on a subset of pseudoknots motifs, the simple-type pseudoknots, and proposed algorithms of complexity in $O(n^4)$ for

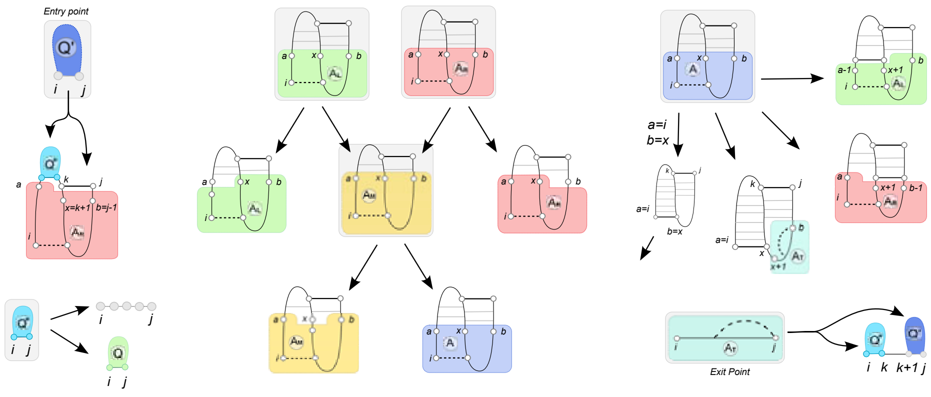


Fig. 4. An unambiguous decomposition for simple non-recursive pseudoknots that captures the Akutsu/Uemura class of pseudoknots. This decomposition yields $O(n^4)/\Theta(n^4)$ time/memory algorithms for partially recursive pseudoknots and can be extended to include recursive pseudoknots and/or Turner energy contributions in $O(n^5)/\Theta(n^4)$.

simple non-recursive pseudoknots in a basepair-maximisation energy model, and in $O(n^5)$ for recursive pseudoknots and loop-based energy models. However, the decomposition proposed in (1) is **ambiguous**, e.g. there exists different ways to create unpaired regions. Therefore we propose in Figure 4 an unambiguous decomposition for the same conformation space.

Previous results. In a previous work (47; 48), one of the authors showed that simple-type pseudoknots can be encoded by a simple formal language, in bijection with a context-free language. Here we focus on partly recursive simple pseudoknots presented in Figure 4. They can be encoded by a well-parenthesized word p over two systems of parentheses $\{(f,\bar{f}),(g,\bar{g})\}$, respectively indicating the leftmost and rightmost basepairs in Figure 4, and an unpaired character c such that

$$p = (c^* f)^n\, p'\, (g\, c^*)^{m_1}\, (\bar{f}\, c^*)^{n_1}\, (g\, c^*)^{m_2}\, (\bar{f}\, c^*)^{n_2} \cdots (g\, c^*)^{m_k}\, (\bar{f}\, c^*)^{n_k - 1}\, \bar{f}\, p''\, \bar{g}\, (c^* \bar{g})^{m-1} \tag{12}$$

where k is some integral value, $\sum_{i=1}^{k} n_i = n \geq 1$, $\sum_{i=1}^{k} m_i = m \geq 1$, and p', p'' are any two recursively-generated conformations.

Completeness. Let us show that the decomposition in Figure 4 is complete, i.e. that any partially recursive pseudoknot can be generated by the decomposition.

Let us initially focus on base-pairs and ignore unpaired bases. The smallest word within the language of Equation 12 is $f p' g \bar{f} p'' \bar{g}$ which can be generated by applying the initial case ($Q \to A_L \to A_M \to A \rightsquigarrow p' \ldots g \ldots \bar{g}$) followed directly by the terminal case ($A \to A_T \rightsquigarrow f\, p'\, g\, \bar{f}\, p''\, \bar{g}$). Moreover through a sequence $A \to A_R \to A_M \to A$, one adds an outermost edge around the right part $g \ldots \bar{g}$. So through m iterations of the sequence the decomposition generates any structure $g^{m_1} \dots \bar{g}^{m_1}$. Similarly through a sequence $A \to A_L \to A_M \to A$ one adds an outermost edge around the left part $f \dots \bar{f}$, and after n_1 iterations any structure $f^{n_1} \dots \bar{f}^{n_1}$ is generated. Since these two sequences can be combined and alternated (starting with the initial case and finishing with the terminal case), then the decomposition generates any word

$$p = f^n\, p'\, g^{m_1}\, \bar{f}^{n_1}\, g^{m_2}\, \bar{f}^{n_2} \cdots g^{m_k}\, \bar{f}^{n_k}\, p''\, \bar{g}^m \bar{g}. \tag{13}$$

For the recursive call p', it is easily verified that Q^* generates any (PK) structure. For p'' it is worth mentioning that, at a base-pairing level, $A \to A_T$ (right base paired) and $A \to \emptyset$ cover all possible situations.

Arbitrary numbers of unpaired bases c can also be inserted right before the opening f of a leftward base pair (resp. after closure $\bar{f}$ of a leftward base pair, after the opening g of a right base pair and before the closure $\bar{g}$ of a right base pair) by repeatedly applying the $A_L \to A_L$ (resp. $A_M \to A_M$, $A_L \to A_L$ and $A_M \to A_M$) rule after adding a left (resp. right) base pair. Consequently any structure described by a word in Equation 12 can be generated by the decomposition.

Unambiguity. Let us now address the unambiguity of the decomposition, using our approach based on generating functions. Equation 12 immediately gives a system of equations relating $AU(z)$, the generating function of simple partially recursive pseudoknots, to $S(z)$ the gen. fun. of all structures:

$$AU(z) = \sum_{k\geq 1} \left(\frac{z}{1-z}\right)^n S(z) \left(\frac{z}{1-z}\right)^{m_1} \left(\frac{z}{1-z}\right)^{n_1} \cdots \left(\frac{z}{1-z}\right)^{n_k-1} z\,S(z)\,z \left(\frac{z}{1-z}\right)^{m-1} = \frac{z^4\, S(z)^2\, (1-z)}{1-2z-z^2}.$$

Now consider the dynamic programming decomposition illustrated by Figure 4. Associating generating functions to each type of vertices and translating assigned bases into monomials, we obtain the following system of equations:

$$Q'(z) = z^2\, S(z)\, A_R(z) \qquad A_L(z) = z\, A_L(z) + A_M(z) \qquad A_R(z) = z\, A_R(z) + A_M(z)$$
$$A_M(z) = z\, A_M(z) + A(z) \qquad A(z) = z^2\, A_R(z) + z^2\, A_L(z) + z^2\, S(z) \qquad A_T(z) = S(z)(1-z) - 1.$$

The last expression for $A_T(z)$ follows directly from the observation that any structure in Q can be written as a sequence of structures from Q' interleaved with sequences of unpaired bases. Given that A_T cannot feature unpaired bases on its right end, one of the sequence of unpaired base must be removed. Furthermore A_T does not generate the empty structure, so we have $S(z) = (A_T(z)+1)/(1-z)$. Solving the system gives $Q'(z) = \frac{S^2(z)\, z^4\, (1-z)}{1-2\,z+z^2} = AU(z)$ and the unambiguity/correctness of the decomposition directly follow.

4.4 Fully-Recursive Kissing Hairpins

Kissing hairpins (KH) are pseudoknotted structure composed of two helices whose terminal loops are linked by a third helix. These pseudoknots are frequently observed, and are exhaustively predicted by Chen *et al* ([8]) in time

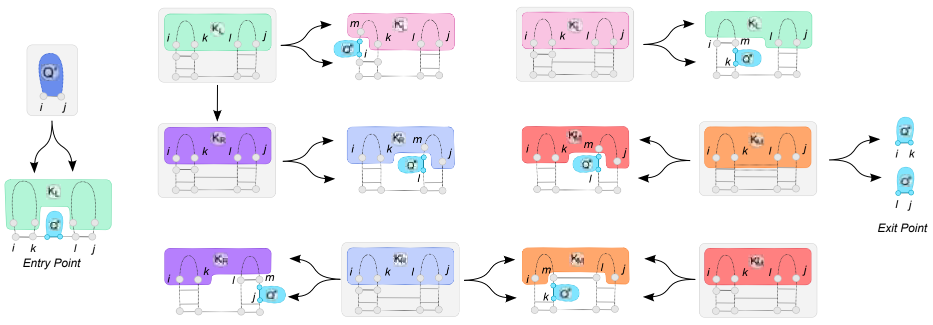


Fig. 5. Unambiguous decomposition of fully recursive kissing hairpins

complexity in $O(n^5)$, and in $O(n^3)/O(n^4)$ under restrictions by Theis *et al* ([50]). Figure 5 presents an unambiguous decomposition which generates the space of recursive kissing hairpins.

Previous results. Again, an encoding of kissing hairpins can be found in earlier work by one of the authors ([47]), showing that any KH pseudoknot can be represented by a word p over three systems of parentheses $\{(f,\bar{f}),(g,\bar{g}),(h,\bar{h})\}$ (respectively denoting leftmost, central and rightmost helices) such that:

$$p = (fS)^n\,(gS)^m\,(\bar{f}S)^n\,(hS)^k\,(\bar{g}S)^m\,(\bar{h}S)^{k-1}\,\bar{h}. \tag{14}$$

Completeness. First let us remark that the *minimal* conformation generated by the decomposition is $K_L \to K_R \to K'_R \to K_M \rightsquigarrow fSgS\bar{f}ShS\bar{g}S\bar{h}$. Remark that one can iterate arbitrarily over the states $K_L \to K'_L \to K_L$, $K'_R \to K_R \to K'_R$ and $K'_M \to K_M \to K_M$. Consequently one may *insert* patterns $(K_L \to K'_L \to K_L)^{n-1} \rightsquigarrow (S\,f)^{n-1}\cdots(\bar{f}\,S)^{n-1}$, $(K'_R \to K_R \to K'_R)^{k-1} \rightsquigarrow (h\,S)^{k-1}\cdots(\bar{h}\,S)^{k-1}$ and $(K_M \to K'_M \to K_M)^m \rightsquigarrow (g\,S)^{m-1}\cdots(S\,\bar{g})^{m-1}$ in the minimal word above, and produce any conformation denoted by

$$f(Sf)^{n-1}S(gS)^{m-1}yS(\bar{f}S)^{n-1}\bar{f}ShS(hS)^{k-1}\bar{g}(S\bar{g})^{m-1}S(\bar{h}S)^{k-1}\bar{h}$$

where one recognizes the language of Equation 14 upon simple expansion.

Unambiguity. Equation 14 allows to derive the generating function $KH(z)$ of kissing-hairpin as a function of $S(z)$ the gen. fun. of all structures:

$$KH(z) = \sum_{n,m,k\geq 1} (zS(z))^n (zS(z))^m (zS(z))^n (zS(z))^k (zS(z))^m (zS(z))^{k-1} z = \frac{z^6 S(z)^5}{(1-z^2S(z)^2)^3}. \tag{15}$$

Now consider the dynamic programming decomposition illustrated by Figure 5, and translate it into a system of functional equation:

$$K(z) = z^4 K_L(z)S(z)$$

$$K_L(z) = S(z)K'_L(z) + K_R(z) \qquad K'_L(z) = z^2 K_L(z)S(z) \qquad K_M(z) = K'_M(z)S(z) + S(z)^2$$

$$K'_M(z) = z^2 K_M(z)S(z) \qquad K_R(z) = K'_R(z)S(z) \qquad K'_R(z) = z^2 K_R(z)S(z) + z^2 K_M(z)S(z)$$

Solving the system gives $K(z) = \frac{z^6 S(z)^5}{(1-z^2 S(z)^2)^3} = KH(z)$ and the unambiguity of the decomposition immediately follows. Again hypergraphs algorithms can be used, and specialize into the complexities summarized in Table 2.

Table 2. Summary of ensemble based algorithms on simple pseudoknots and kissing hairpins. π_{bp} stands for the simple Nussinov-Jacobson energy model, and $\pi_{\mathcal{T}}$ for a Turner-like model based on loops contributions.

Application	Algorithm	Weight fun.	Time	Memory	Ref.
Simple type pseudoknots (Akutsu&Uemura)					
A – Energy minimization	Minimal weight	π_{bp}	$O(n^4)$	$O(n^4)$	(1)
B – Partition function	Weighted count	$e^{\frac{-\pi_{bp}}{RT}}$	$O(n^4)$	$O(n^4)$	(6; 7) in $\Theta(n^6)$
C – Base-pairing probabilities	Arc-traversal prob.	$e^{\frac{-\pi_{bp}}{RT}}$	$O(n^4)$	$O(n^4)$	–
D – Statistical sampling (k-samples)	Weighted rand. gen.	$e^{\frac{-\pi_{bp}}{RT}}$	$O(n^4 + k \cdot n \log n)$	$O(n^4)$	–
E – Moments of energy (Mean, Var.)	Moments extraction	$e^{\frac{-\pi_{bp}}{RT}}$	$O(n^4)$	$O(n^4)$	–
F – m-th moment of additive features	Moments extraction	$e^{\frac{-\pi_{bp}}{RT}}$	$O(m^3 \cdot n^4)$	$O(m \cdot n^4)$	–
Fully recursive Kissing Hairpins					
A – Energy minimization	Minimal weight	$\pi_{\mathcal{T}}$	$O(n^5)$	$O(n^4)$	(8)
B – Partition function	Weighted count	$e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$O(n^5)$	$O(n^4)$	–
C – Base-pairing probabilities	Arc-traversal prob.	$e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$O(n^5)$	$O(n^4)$	–
D – Statistical sampling (k-samples)	Weighted rand. gen.	$e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$O(n^5 + k \cdot n \log n)$	$O(n^4)$	–
E – Moments of energy (Mean, Var.)	Moments extraction	$e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$O(n^5)$	$O(n^4)$	–
F – m-th moment of additive features	Moments extraction	$e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$O(m^3 \cdot n^5)$	$O(m \cdot n^4)$	–

5 Extending the Framework: Extraction of Moments and Exact Correlations

A last application addresses the extraction of statistical measures for **additive features**. Let us first define a **feature** as a function $\alpha : E \to \mathbb{R}^+$ extended additively over F-paths such that $\alpha(p) = \sum_{e \in p} \alpha(e)$. One may then want to characterize the distribution of a random variable $X = \alpha(p)$, for $p \in \mathcal{P}$ a random F-path drawn according to the weighted distribution. As it is not necessarily feasible to determine the exact distribution of X, one can examine statistical measures such as its

$$\text{Mean } \mu_X = \mathbb{E}[X] \quad \text{and} \quad \text{Variance } \mathrm{Var}_X = \mathbb{E}[X^2] - \mu_X^2,$$

e.g. from which the distribution is fully determined in the case of Gaussian distributions. Even when the distribution is not normal, it can still be characterized by a list of measures called **moments** of X, the m-th moment being defined as $\mathbb{E}[X^m] = \sum_{p \in \mathcal{P}} \alpha(p)^m \cdot \pi(p)/w_s$.

Moreover in the presence of multiple features $(X_1 := \alpha_1(p), \ldots, X_k := \alpha_k(p))$, similar measures can be used to estimate their level of dependency. One such measure is the **Pearson product-moment correlation coefficient** ρ_{X_1,X_2} defined for two random variables as

$$\rho_{X_1,X_2} = \frac{\mathrm{Cov}_{X_1,X_2}}{\sqrt{\mathrm{Var}_{X_1} \cdot \mathrm{Var}_{X_2}}} = \frac{\mathbb{E}[X_1 \cdot X_2] - \mathbb{E}[X_1] \cdot \mathbb{E}[X_2]}{\sqrt{\mathrm{Var}_{X_1} \cdot \mathrm{Var}_{X_2}}}$$

The correlation above involves the expectation of a product of two random variables which is an instance of a **generalized moment**, defined for the set of F-paths starting from $s \in V$ as

$$\mathbb{E}[X_1^{m_1} \cdots X_k^{m_k} \mid s] = \sum_{p \in \mathcal{P}_s} \frac{\pi(p)}{w_s} \prod_{i=1}^{k} \alpha_i(p)^{m_i}. \tag{16}$$

Extracting such moments can be quite useful, allowing one to get access to average properties of structures (#Hairpins, #Occurrences of pseudoknots...) and their correlations within a weighted ensemble. For instance, Miklos *et al* ([38]) proposed an $\mathcal{O}(m^2 \cdot n^3)$ algorithm for computing the m-th moment of the Energy distribution for secondary structure in order to compare the distribution of free-energy in non-coding RNAs and random sequences. We are going to show how these generalized moments can be extracted directly through a generalization of the weighted count algorithm.

Theorem 1. *Let* $\alpha := (\alpha_1, \cdots, \alpha_k)$ *be a vector of additive features and* $\mathbf{m} := (m_1, \cdots, m_k)$ *be a k-tuple of natural integers. Then the pseudo-moment* $c_s^{\mathbf{m}} := \mathbb{E}[X_1^{m_1} \cdots X_k^{m_k} \mid s] \cdot w_s$ *of* α *in a weighted distribution can be recursively computed through*

$$c_s^{\mathbf{m}} = \sum_{e=(s \to \mathbf{t})} \pi(e) \cdot \sum_{\substack{\mathbf{m}', (\mathbf{m}''_1, \cdots, \mathbf{m}''_{|t|}) \\ s.\ t.\ \mathbf{m}' + \sum_j \mathbf{m}''_j = \mathbf{m}}} \prod_{i=1}^{k} \binom{m_i}{m'_i, m''_{1,i}, \cdots, m''_{|t|,i}} \cdot \alpha_i(e)^{m'_i} \cdot \prod_{i=1}^{|t|} c_{t_i}^{\mathbf{m}''_i} \tag{17}$$

in $\mathcal{O}\left((|E| + |V|) \cdot k \cdot t^+ \cdot \prod_{i=1}^{k} m_i^{t^+ + 1}\right)$ *time complexity and* $\Theta\left(|V| \cdot \prod_{i=1}^{k} m_i\right)$ *memory where* $t^+ = \max_{(s \to t) \in E}(|t|)$ *is the maximal out-degree of an arc.*

Adding this new generic algorithms automatically creates new applications for each an every conformation space as summarized in Figure 2. This simultaneous extension – for all conformational spaces – of possible ensemble applications constitues in our opinion one of the main benefit of detaching the decomposition from its exploration.

6 Conclusion and Perspectives

In this paper, we established the foundation of a combinatorial approach to the design of algorithms for complex conformation spaces. We built on an hypergraph model introduced in the context of RNA secondary structure by Finkelstein and Roytberg ([16]), which we extended in several direction. First we formulated classic and novel generic algorithms on Forward-Hypergraphs for weighted ensembles, allowing one to derive base-pairing probabilities, perform statistical sampling and extract moments of the distribution of additive features. Then we showed how combinatorial arguments based on generating functions could be used to simplify the proof of correctness for designed decompositions. We illustrated the full programme on classic secondary structures, simple type pseudoknots and fully-recursive kissing hairpin pseudoknots for which we provided decompositions that were proven to be unambiguous and complete with respect to previous work. The hypergraph formulation of the decomposition, coupled with the generic algorithms, readily gave a family of novel algorithms for complex – yet relevant – conformation spaces.

Let us mention some perspectives to our contribution. Firstly the principles and algorithms described here could easily be implemented as a general *compiler* tools for F-Graphs algorithms. Such a compiler could be coupled with helper tools expanding hypergraphs from succinct descriptions, such as context-free grammars (related to ADP ([19])), or M. Möhl's split types ([37]). More complex search space could also be modeled, such as those relying on a more detailed representation of RNA structure (e.g. MCFold's NCMs ([40])), those capturing RNA-RNA interactions ([2; 24]), those offering simultaneous alignment and folding (Sankoff's algorithm ([46])) or performing mutations on the sequence ([53]). Finally our hypergraph framework is not necessarily limited to polynomial algorithms, and algorithmic developments could be proposed to address some of the current algorithmic issues in RNA (inverse folding ([3]), kinetics ([49])) for which no exact polynomial algorithms are currently known (or suspected). More generally it is our hope that, by simplifying and modularizing the process of developing new – algorithmically tractable – conformation spaces, our contribution will help design better, more topologically-realistic([52; 28; 44]), energy and conformational spaces to better understand and predict the structure(s) of RNA.

Acknowledgement. The authors wish to express their gratitude to M. Roytberg for pointing out his work on hypergraphs as a unifying framework, and to R. Backofen, M. Möhl and S. Will for fruitful discussions. This research was supported by the Digiteo project "RNAomics". YP was funded by an ANR grant MAGNUM (ANR 2010 BLAN 0204).

References

[1] Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discrete Appl. Math. 104(1-3), 45–62 (2000)

[2] Alkan, C., Karakoç, E., Nadeau, J.H., Şahinalp, S.C., Zhang, K.: RNA-RNA Interaction Prediction and Antisense RNA Target Search. In: Miyano, S., Mesirov, J., Kasif, S., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) RECOMB 2005. LNCS (LNBI), vol. 3500, pp. 152–171. Springer, Heidelberg (2005)

[3] Andronescu, M., Fejes, A.P., Hutter, F., Hoos, H.H., Condon, A.: A New Algorithm for RNA Secondary Structure Design. J. Mol. Biol. 336(3), 607–624 (2004)

[4] Bekaert, M., Bidou, L., Denise, A., Duchateau-Nguyen, G., Forest, J., Froidevaux, C., Hatin, I., Rousset, J., Termier, M.: Towards a computational model for – 1 eukaryotic frameshifting sites. Bioinformatics 19, 327–335 (2003)

[5] Bousquet-Mélou, M., Ponty, Y.: Culminating paths. Discrete Mathematics and Theoretical Computer Science 10(2), 125–152 (2008)
[6] Cao, S., Chen, S.J.: Predicting RNA pseudoknot folding thermodynamics. Nucleic Acids Res. 34(9), 2634–2652 (2006)
[7] Cao, S., Chen, S.J.: Predicting structured and stabilities for H-type pseudoknots with interhelix loop. RNA 15, 696–706 (2009)
[8] Chen, H.L., Condon, A., Jabbari, H.: An O(n(5)) algorithm for MFE prediction of kissing hairpins and 4-chains in nucleic acids. Journal of Computational Biology 16(6), 803–815 (2009)
[9] Condon, A., Davy, B., Rastegari, B., Zhao, S., Tarrant, F.: Classifying RNA pseudoknotted structures. Theoretical Computer Science 320(1), 35–50 (2004)
[10] Denise, A., Ponty, Y., Termier, M.: Controlled non uniform random generation of decomposable structures. Theoretical Computer Science 411(40-42), 3527–3552 (2010)
[11] Ding, Y., Chan, C.Y., Lawrence, C.E.: RNA secondary structure prediction by centroids in a boltzmann weighted ensemble. RNA 11, 1157–1166 (2005)
[12] Ding, Y., Lawrence, E.: A statistical sampling algorithm for RNA secondary structure prediction. Nucleic Acids Res. 31(24), 7280–7301 (2003)
[13] Dirks, R., Pierce, N.: A partition function algorithm for nucleic acid secondary structure including pseudoknots. J. Comput. Chem. 24, 1664–1677 (2003)
[14] Do, C.B., Woods, D.A., Batzoglou, S.: CONTRAfold: RNA secondary structure prediction without physics-based models. Bioinformatics 22(14), e90–e98 (2006)
[15] Ferrè, F., Ponty, Y., Lorenz, W.A., Clote, P.: DIAL: A web server for the pairwise alignment of two RNA 3-dimensional structures using nucleotide, dihedral angle and base pairing similarities. Nucleic Acids Res. 35 (Web server issue), W659–W668 (July 2007)
[16] Finkelstein, A.V., Roytberg, M.A.: Computation of biopolymers: a general approach to different problems. Biosystems 30(1-3), 1–19 (1993)
[17] Flajolet, P., Zimmermann, P., Van Cutsem, B.: Calculus for the random generation of labelled combinatorial structures. Theoretical Computer Science 132, 1–35 (1994), a preliminary version is available in INRIA Research Report RR-1830
[18] Flajolet, P.: Analytic models and ambiguity of context-free languages. Theoretical Computer Science 49, 283–309 (1987)
[19] Giegerich, R.: A systematic approach to dynamic programming in bioinformatics. Bioinformatics 16(8), 665–677 (2000)
[20] Hamada, M., Kiryu, H., Sato, K., Mituyama, T., Asai, K.: Prediction of RNA secondary structure using generalized centroid estimators. Bioinformatics 25(4), 465–473 (2009)
[21] Harmanci, A.O., Sharma, G., Mathews, D.H.: Stochastic sampling of the rna structural alignment space. Nucleic Acids Res. 37(12), 4063–4075 (2009)
[22] Hofacker, I.L.: Vienna RNA secondary structure server. Nucleic Acids Res. 31(13), 3429–3431 (2003)
[23] Huang, F.W.D., Peng, W.W.J., Reidys, C.M.: Folding 3-noncrossing rna pseudoknot structures. J. Comput. Biol. 16(11), 1549–1575 (2009)
[24] Huang, F.W.D., Qin, J., Reidys, C.M., Stadler, P.F.: Target prediction and a statistical sampling algorithm for RNA-RNA interaction. Bioinformatics 26(2), 175–181 (2010)
[25] Kucherov, G., Noe, L., Ponty, Y.: Estimating seed sensibility on homogenous alignments. In: IEEE (ed.) Proceedings of Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE 2004), p. 387 (2004)

[26] Lefebvre, F.: A grammar-based unification of several alignment and folding algorithms. In: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, pp. 143–154. AAAI Press, Menlo Park (1996)

[27] Lefebvre, F.: Grammaires S-attribuées multi-bandes et applications à l'analyse automatique de séquences biologiques. Ph.D. thesis, École Polytechnique (1997)

[28] Lescoute, A., Westhof, E.: Topology of three-way junctions in folded RNAs. RNA 12(1), 83–93 (2006)

[29] Lorenz, W., Ponty, Y., Clote, P.: Asymptotics of RNA shapes. Journal of Computational Biology 15(1), 31–63 (2008)

[30] Lyngsø, R.B., Pedersen, C.N.S.: RNA pseudoknot prediction in energy-based models. Journal of Computational Biology 7(3-4), 409–427 (2000)

[31] Markham, N.R.: Algorithms and software for nucleic acid sequences. Ph.D. thesis, Faculty of Rensselaer Polytechnic Institute (2006)

[32] Markham, N.R., Zuker, M.: UNAFold: software for nucleic acid folding and hybridization. Methods Mol. Biol. 453, 3–31 (2008)

[33] Mathews, D.H.: Using an RNA secondary structure partition function to determine confidence in base pairs predicted by free energy minimization. RNA 10(8), 1178–1190 (2004)

[34] Mathews, D., Sabina, J., Zuker, M., Turner, D.: Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. J. Mol. Biol. 288, 911–940 (1999)

[35] McCaskill, J.: The equilibrium partition function and base pair binding probabilities for RNA secondary structure. Biopolymers 29, 1105–1119 (1990)

[36] Mückstein, U., Hofacker, I.L., Stadler, P.F.: Stochastic pairwise alignments. Bioinformatics 18(suppl. 2), S153–S160 (2002)

[37] Möhl, M., Will, S., Backofen, R.: Lifting prediction to alignment of rna pseudoknots. J. Comput. Biol. 17(3), 429–442 (2010), `http://dx.doi.org/10.1089/cmb.2009.0168`

[38] Miklós, I., Meyer, I.M., Nagy, B.: Moments of the boltzmann distribution for RNA secondary structures. Bull. Math. Biol. 67(5), 1031–1047 (2005)

[39] Nussinov, R., Jacobson, A.B.: Fast algorithm for predicting the secondary structure of single stranded RNA. Proc. Natl. Acad. Sci. USA 77(11), 6309–6313 (1980)

[40] Parisien, M., Major, F.: The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data. Nature 452(7183), 51–55 (2008)

[41] Ponty, Y.: Efficient sampling of RNA secondary structures from the boltzmann ensemble of low-energy: The boustrophedon method. J. Math. Biol. 56(1-2), 107–127 (2008)

[42] Reeder, J., Giegerich, R.: Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. BMC Bioinformatics 5, 104 (2004)

[43] Reeder, J., Steffen, P., Giegerich, R.: Effective ambiguity checking in biosequence analysis. BMC Bioinformatics 6, 153 (2005)

[44] Reidys, C.M., Huang, F.W.D., Andersen, J.E., Penner, R.C., Stadler, P.F., Nebel, M.E.: Topology and prediction of rna pseudoknots. Bioinformatics 27(8), 1076–1085 (2011)

[45] Rivas, E., Eddy, S.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. J. Mol. Biol. 285, 2053–2068 (1999)

[46] Sankoff, D.: Simultaneous solution of the rna folding, alignment and protosequence problems. SIAM J. Appl. Math. 45, 810–825 (1985)

[47] Saule, C.: Modèles combinatoires des structures d'ARN avec ou sans pseudo-noeuds, application à la comparaison de structures. Ph.D. thesis, Université Paris Sud, Ecole doctorale informatique (December 2010)

[48] Saule, C., Régnier, M., Steyaert, J.M., Denise, A.: Counting RNA pseudoknotted structures. Journal of Computational Biology (to appear)

[49] Thachuk, C., Manuch, J., Rafiey, A., Mathieson, L.A., Stacho, L., Condon, A.: An algorithm for the energy barrier problem without pseudoknots and temporary arcs. In: Pac. Symp. Biocomput., pp. 108–119 (2010)

[50] Theis, C., Janssen, S., Giegerich, R.: Prediction of RNA secondary structure including kissing hairpin motifs. In: Moulton, V., Singh, M. (eds.) WABI 2010. LNCS, vol. 6293, pp. 52–64. Springer, Heidelberg (2010)

[51] Tinoco, I., Borer, P.N., Dengler, B., Levin, M.D., Uhlenbeck, O.C., Crothers, D.M., Bralla, J.: Improved estimation of secondary structure in ribonucleic acids. Nat. New. Biol. 246(150), 40–41 (1973)

[52] Vernizzi, G., Ribeca, P., Orland, H., Zee, A.: Topology of pseudoknotted homopolymers. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 73(3), 031902 (2006)

[53] Waldispühl, J., Devadas, S., Berger, B., Clote, P.: Efficient algorithms for probing the RNA mutation landscape. PLoS Comput Biol 4(8), e1000124 (2008)

[54] Waterman, M.S.: Secondary structure of single stranded nucleic acids. Advances in Mathematics Supplementary Studies 1(1), 167–212 (1978)

[55] Wilf, H.S.: A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. Advances in Mathematics 24, 281–291 (1977)

[56] Zuker, M., Stiegler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Res. 9, 133–148 (1981)

Indexing Finite Language Representation of Population Genotypes

Jouni Sirén[⋆], Niko Välimäki[⋆⋆], and Veli Mäkinen

Helsinki Institute for Information Technology (HIIT) &
Department of Computer Science, University of Helsinki, Finland
{jltsiren,nvalimak,vmakinen}@cs.helsinki.fi

Abstract. We propose a way to index population genotype information together with the complete genome sequence, so that one can use the index to efficiently align a given sequence to the genome with all plausible genotype recombinations taken into account. This is achieved through converting a multiple alignment of individual genomes into a finite automaton recognizing all strings that can be read from the alignment by switching the sequence at any time. The finite automaton is indexed with an extension of Burrows-Wheeler transform to allow pattern search inside the plausible recombinant sequences. The size of the index stays limited, because of the high similarity of individual genomes. The index finds applications in variation calling and in primer design.

1 Introduction

Due to the advances in DNA sequencing [17], it is now possible to have complete genomes of individuals sequenced and assembled. Already several human genomes have been sequenced [22,9,11,23,14] and it is almost a routine task to resequence individuals by aligning the high-throughput short DNA reads to the reference [7]. This rich and focused genotype information, together with the more global genotype information (common single nucleotide polymorphisms (SNPs) and other variations) created using earlier techniques, can be combined to do different population-wide studies, now first time directly on whole genome level.

We propose a novel index structure that simultaneously represents and extrapolates the genotype information present in the population samples. The index structure is built on a given multiple alignment of individual genomes, or alternatively for a single reference sequence and set of SNPs of interest. The index structure is capable of aligning a given pattern to any path taken along the multiple alignment, as illustrated in Figure 1.

To build the index, we first create a finite automaton recognizing all paths through the multiple alignment, and then generalize Burrows-Wheeler transform

⋆ Funded by the Finnish Doctoral Programme in Computational Sciences, Academy of Finland project 1140727 and the Nokia Foundation.

⋆⋆ Funded by Helsinki Graduate School in Computer Science and Engineering and Finnish Centre of Excellence for Algorithmic Data Analysis Research

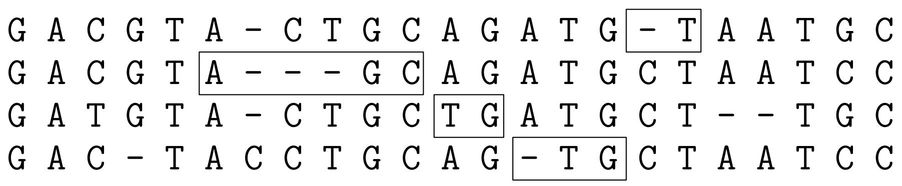

G A C G T A - C T G C A G A T G - T A A T G C
G A C G T A - - - G C A G A T G C T A A T C C
G A T G T A - C T G C T G A T G C T - - T G C
G A C - T A C C T G C A G - T G C T A A T C C

Fig. 1. Pattern `AGCTGTGT` matching the multiple alignment when allowing it to change row when necessary

(BWT) [2] -based self-index structures [19] to index paths in labeled graphs. The backward search routine of BWT-based indexes generalizes to support exact pattern search over the labeled graph in $O(m)$ time, for pattern of length m. On general labeled graphs, such index can take exponential space, but on graphs resulting from finite automaton representation of multiple alignment of individual genomes, the space is expected to stay limited.

Applications for our index include the following:

– *SNP calling.* We can take the known SNPs into account already in the short read alignment phase, instead of the common pipeline of alignment, variation calling, and filtering of known SNPs. This allows more accurate alignment, as the known SNPs are no longer counted as errors, and the matches can represent novel recombinants not yet represented in the database.
– *Probe/primer design.* When designing probes for microarrays or primers for PCR, it is important that the designed sequence does not occur even approximately elsewhere than in the target. Our index can provide approximate search not only against all substrings, but also against plausible recombinants, and hence the design can be made more selective.
– *Large indel calling.* After short read alignment, the common approach for detecting larger indels is to study the uncovered regions in the reference genome. We can model deletions with our index by adding an edge to the automaton skipping each plausible deleted area. For insertions, we can apply *de novo* sequence assembly with the unmapped reads to generate plausible insertions, and add these as paths to the automaton.
– *Reassembly of donor genome.* Continuing from variation calling, one can use the realignment of the reads to the refined automaton to give a probability for each edge. It is then easy to extract, for example, the most probable path through the automaton as a consensus for the donor.

We made some feasibility experiments on SNP calling problem. We created our index on a multiple alignment of four instances of the 76 Mbp human chromosome 18. The total size of the index was about 67 MB. Aligning a set of 10 million Illumina Solexa reads of length 56 took 18 minutes, and about 1.1% of exact matches were novel recombinants not found when indexing each chromosome instance separately (see Sect. 5). Leaving these exact matches out from variation calling reduces the number of novel SNPs from 4203 to 1074.

Related work and extensions. *Jumping alignments* of protein sequences were studied in [21] as an alternative to profile Hidden Markov Models. They showed how to do local alignment across a multiple alignment of protein family so that jumping from one sequence to another is associated with a penalty. We study the same problem but from a different angle; we provide a compressed representation of the multiple alignment with an efficient way to support pattern matching.

Calling of large indels similar to our approach was studied in [1]. One difference is that they manage to associate probabilities to the putative genotypes, resulting into a more reliable calling of likely variants. However, their indexing part is tailored for this specific problem, whereas we develop a more systematic approach that can be generalized to many directions. For example, we can take the probabilities into account and index only paths with high enough probabilities, to closely simulate their approach (see Sect. 6).

Our work builds on the self-indexing scenario [19], and more specifically is an extension of the *XBW transform* [5] that is an index structure for labeled trees. Our extension to *labeled graphs* may be of independent interest, as it has potentially many more applications inside and outside computational biology.

The focus of this paper is the finite automaton representation of a multiple alignment. This setting is closely related to our previous work on indexing highly repetitive sequence collections [15]. In our previous work, we represented a collection of individual genomes of total length N, with reference sequence of length n, and a total of s mutations, in space $O(n \log \frac{N}{n} + s \log^2 N)$ bits in the average case (rough upper bounds here for simplicity). Exact pattern matching was supported in $O(m \log N)$ time. The index proposed in this paper achieves $O(n(1 + s/n)^{O(\log n)})$ bits in the expected case for constant-sized alphabets.

The paper is organized as follows. Section 2 introduces the notation, Sect. 3 reviews the necessary index structures we build on, Sect. 4 describes the new extension to finite languages, Sect. 5 gives some preliminary experiments on SNP calling problem, and Sect. 6 concludes the work by sketching the steps required for making the index into a fully applicable tool.

2 Definitions

A *string* $S = S[1,n]$ is a *sequence* of *characters* from *alphabet* $\Sigma = \{1, 2, \ldots, \sigma\}$. A *substring* of S is written as $S[i,j]$. A substring of type $S[1,j]$ is called a *prefix*, while a substring of type $S[i,n]$ is called a *suffix*. A *text* string $T = T[1,n]$ is a string terminated by $T[n] = \$ \notin \Sigma$ with lexicographic value 0. The *lexicographic order* "$<$" among strings is defined in the usual way.

A *graph* $G = (V, E)$ consists of a set $V = \{v_1, \ldots, v_{|V|}\}$ of *nodes* and a set $E \subset V^2$ of *edges* such that $(v, v) \notin E$ for all $v \in V$. We call $(u, v) \in E$ an edge from node u to node v. A graph is *directed*, if edge (u, v) is distinct from edge (v, u). In this paper, the graphs are always directed. For every node $v \in V$, the *indegree* $in(v)$ is the number of incoming edges (u, v), and the *outdegree* $out(v)$ the number of outgoing edges (v, w).

Graph G is said to be *labeled*, if we attach a *label* $\ell(v)$ to each node $v \in V$. A *path* $P = u_1 \cdots u_{|P|}$ is a sequence of nodes from u_1 to $u_{|P|}$ such that $(u_i, u_{i+1}) \in$

E for all $i < |P|$. The label of path P is the string $\ell(P) = \ell(u_1) \cdots \ell(u_{|P|})$. A *cycle* is a path from a node to itself containing, at least one other node. If a graph contains no cycles, it is called *acyclic*.

A *finite automaton* is a directed labeled graph $A = (V, E)$[1]. The *initial node* v_1 is labeled with $\ell(v_1) = \#$ with lexicographic value $\sigma + 1$, while the *final node* $v_{|V|}$ is labeled with $\ell(v_{|V|}) = \$$. The rest of the nodes are labeled with characters from alphabet Σ. Every node is assumed to be on some path from v_1 to $v_{|V|}$.

The *language* $L(A)$ recognized by automaton A is the set of the labels of all paths from v_1 to $v_{|V|}$. If the language contains a finite number of strings, it is called *finite*. A language is finite if and only if the automaton is acyclic. Two automata are said to be *equivalent*, if they recognize the same language.

Automaton A is forward (reverse) *deterministic* if, for every node $v \in V$ and every character $c \in \Sigma \cup \{\#, \$\}$, there exists at most one node u such that $\ell(u) = c$ and $(v, u) \in E$ ($(u, v) \in E$). For any language recognized by some finite automaton, we can always construct an equivalent automaton that is forward (reverse) deterministic.

3 Compressed Indexes

The *suffix array (SA)* of text $T[1, n]$ is an array of pointers $\mathsf{SA}[1, n]$ to the suffixes of T in lexicographic order. As an abstract data type, a suffix array is any data structure that supports the following operations efficiently: (a) *find* the SA range containing the suffixes prefixed by *pattern* P; (b) *locate* the suffix $\mathsf{SA}[i]$ in the text; and (c) *display* any substring of text T.

Compressed suffix arrays (CSA) [8,6] support these operations. Their compression is based on the *Burrows-Wheeler transform (BWT)* [2], a permutation of the text related to the SA. The BWT of text T is a sequence $\mathsf{BWT}[1, n]$ such that $\mathsf{BWT}[i] = T[\mathsf{SA}[i] - 1]$, if $\mathsf{SA}[i] > 1$, and $\mathsf{BWT}[i] = T[n] = \$$ otherwise.

BWT can be reversed by a permutation called *LF-mapping* [2,6]. Let $C[1, \sigma]$ be an array such that $C[c]$ is the number of characters in $\{\$, 1, 2, \ldots, c - 1\}$ occurring in the BWT. We also define $C[0] = C[\$] = 0$ and $C[\sigma + 1] = n$. We define LF-mapping as $LF(i) = C[\mathsf{BWT}[i]] + rank_{\mathsf{BWT}[i]}(\mathsf{BWT}, i)$, where $rank_c(\mathsf{BWT}, i)$ is the number of occurrences of character c in prefix $\mathsf{BWT}[1, i]$.

The inverse of LF-mapping is $\Psi(i) = select_c(\mathsf{BWT}, i - C[c])$, where c is the highest value with $C[c] < i$, and $select_c(\mathsf{BWT}, j)$ is the position of the jth occurrence of character c in BWT [8]. By its definition, function Ψ is strictly increasing in the range $[C[c]+1, C[c+1]]$ for every $c \in \Sigma$. Note that $T[\mathsf{SA}[i]] = c$ and $\mathsf{BWT}[\Psi(i)] = c$ for $C[c] < i \leq C[c + 1]$.

These functions form the backbone of CSAs. As $\mathsf{SA}[LF(i)] = \mathsf{SA}[i] - 1$ [6] and hence $\mathsf{SA}[\Psi(i)] = \mathsf{SA}[i] + 1$, we can use these functions to move the suffix array position backward and forward in the sequence. The functions can be efficiently implemented by adding some extra information to a compressed representation of the BWT. Standard techniques [19] for supporting SA functionality include

[1] Unlike the usual definition, we label nodes instead of edges.

using *backward searching* [6] for *find*, and sampling some suffix array values for *locate* and *display*.

XBW [5] is a generalization of the Burrows-Wheeler transform for labeled trees, where leaf nodes and internal nodes are labeled with different alphabets. Internal nodes of the tree are sorted lexicographically according to path labels from the node to the root. Sequence BWT is formed by concatenating the labels of the children of each internal node in lexicographic order according to the parent node. Every internal node v now corresponds to a substring $\mathsf{BWT}[sp_v, ep_v]$ containing the labels of its children. The first position sp_v of each such substring is marked with a 1-bit in bit vector F. Backward searching is used to support the analogue of *find*. Tree navigation is possible by using BWT and F.

4 Burrows-Wheeler Transform for Finite Languages

In this section, we generalize the XBW approach to finite automata. We call it the *generalized compressed suffix array (GCSA)*. For the GCSA to function, we require that the automaton is prefix-sorted. Refer to Section 4.4 on how to transform an automaton into an equivalent prefix-sorted automaton.

Definition 1. *Let A be a finite automaton, and let $v \in V$ be a node. Node v is* prefix-sorted *by prefix $p(v)$, if the labels of all paths from v to $v_{|V|}$ share a common prefix $p(v)$, and no path from any other node $u \neq v$ to $v_{|V|}$ has $p(v)$ as a prefix of its label. Automaton A is prefix-sorted, if all nodes are prefix-sorted.*

Every node of a prefix-sorted automaton A corresponds to a lexicographic range of suffixes of language $L(A)$.

In XBW, bit vector F is used to mark both nodes and edges. If node v has lexicographic rank i, the labels of its predecessors are $\mathsf{BWT}[sp_v, ep_v] = \mathsf{BWT}[select_1(F, i), select_1(F, i+1)-1]$. On the other hand, if node u is a child of node v, and $\mathsf{BWT}[j]$ contains the label of node u, then $LF(j)$ is the lexicographic rank of the label of the path from node u through node v to the root. Hence $select_1(F, LF(j))$ gives us the position of edge (u, v).

While the latter functionality is trivial in trees, a node can have many outgoing edges in a finite automaton. Hence we will use another bit vector M to mark the outgoing edges.

Let $A = (V, E)$ be a prefix-sorted automaton. To build GCSA, we sort the nodes $v \in V$ according to prefixes $p(v)$. For every node $v \in V$, sequence BWT and bit vectors F and M contain range $[sp_v, ep_v]$ of length $n(v) = \max(in(v), out(v))$. See Figure 4 and Table 1 for an example.

- $\mathsf{BWT}[sp_v, ep_v]$ contains the labels $\ell(u)$ for all incoming edges $(u, v) \in E$, followed by $n(v) - in(v)$ empty characters.
- $F[sp_v] = 1$ and $F[i] = 0$ for $sp_v < i \leq ep_v$.
- $M[sp_v, ep_v]$ contains $out(v)$ 1-bits followed by $n(v) - out(v)$ 0-bits. For the final node $v_{|V|}$, the range contains one 1-bit followed by 0-bits.

```
function $LF([sp_v, ep_v], c)$:
1  $i \leftarrow C[c] + rank_c(\mathsf{BWT}, ep_v)$
2  if $select_c(\mathsf{BWT}, i - C[c]) < sp_v$:
3    return $\emptyset$
4  $i \leftarrow select_1(M, i)$
5  $sp_u \leftarrow select_1(F, rank_1(F, i))$
6  $ep_u \leftarrow select_1(F, rank_1(F, i) + 1) - 1$
7  return $[sp_u, ep_u]$

function $\ell([sp_v, ep_v])$:
8  return $char(rank_1(M, sp_v))$

function $\Psi([sp_u, ep_u])$:
9  $c \leftarrow \ell([sp_u, ep_u])$
10 $res \leftarrow \emptyset$
11 $low \leftarrow rank_1(M, sp_u)$
12 $high \leftarrow rank_1(M, ep_u)$
13 for $i \leftarrow low$ to $high$:
14   $j \leftarrow select_c(\mathsf{BWT}, i - C[c])$
15   $sp_v \leftarrow select_1(F, rank_1(F, j))$
16   $ep_v \leftarrow select_1(F, rank_1(F, j) + 1) - 1$
17   $res \leftarrow res \cup \{[sp_v, ep_v]\}$
18 return $res$
```

Fig. 2. Pseudocode for the basic navigation functions LF, Ψ, and ℓ

Array C is used with some modifications. We define $C[\sigma + 1] = C[\#]$ in the same way as for regular characters, while $C[\sigma + 2]$ is set to be $|E|$. Assuming that each edge $(u, v) \in E$ has an implicit label $\ell(u)p(v)$, we can interpret $C[c]$ as the number of edges with labels smaller than c. We write $char(i)$ to denote character c such that $C[c] < i \leq C[c + 1]$.

4.1 Basic Navigation

Let $[sp_v, ep_v]$ be the range of BWT corresponding to node $v \in V$. We define the following functions:

– $LF([sp_v, ep_v], c) = [sp_u, ep_u]$, where $\ell(u) = c$ and $(u, v) \in E$, or $\emptyset$ if no such u exists.
– $\Psi([sp_u, ep_u]) = \{[sp_v, ep_v] \mid (u, v) \in E\}$.
– $\ell([sp_v, ep_v]) = \ell(v)$.

These are generalizations of the respective functions on BWT. LF can be used to move backwards on edge (u, v) such that $\ell(u) = c$, while Ψ lists the endpoints of all outgoing edges from node u. These functions can be implemented by using BWT, F, M, and C, as seen in Figure 2.

Line 1 of LF is similar to the regular LF, determining the rank of edge label $cp(v)$ among all edge labels. On lines 2 and 3, we determine if there is an occurrence of c in $\mathsf{BWT}[sp_v, ep_v]$. On line 4, we find the position of edge (u, v) in bit vector M, and on lines 5 and 6, we find the range $[sp_u, ep_u]$ containing this position.

4.2 Searching

As the generalized compressed suffix array is a CSA, most of the algorithms using a CSA can be modified to use GCSA instead. In this section, we describe how to support two of the basic SA operations (see Sect. 3):

– *find*(P) returns the range $[sp, ep]$ of BWT corresponding to those nodes v, where at least one path starting from v has pattern P as a prefix of its label.
– *locate*$([sp_v, ep_v])$ returns a numerical value corresponding to node v.

We use backward searching [6] to support *find*. The algorithm maintains an invariant that $[sp_i, ep_i]$ is the range returned by $\mathit{find}(P[i,m])$. In the initial step, we start with the edge range $[C[P[m]]+1, C[P[m]+1]]$, and convert it to range $[sp_m, ep_m]$ by using bit vectors M and F. The step from $[sp_{i+1}, ep_{i+1}]$ to $[sp_i, ep_i]$ is a generalization of function LF for a range of nodes. We find the first and last occurrences of character $P[i]$ in $\mathsf{BWT}[sp_{i+1}, ep_{i+1}]$, map them to edge ranks by using C and BWT, and convert the ranks to sp_i and ep_i by using F and M.

For *locate*, we assume that there is a (not necessarily unique) numerical value $id(v)$ attached to each node $v \in V$. Examples of these values include node ids (so that $id(v_i) = i$) and positions in the multiple alignment. To avoid excessive sampling of node values, $id(v)$ should be $id(u)+1$ whenever (u,v) is the only outgoing edge from u and the only incoming edge to v.

We sample $id(u)$, if there are multiple or no outgoing edges from node u, or if $id(v) \neq id(u)+1$ for the only outgoing edge (u,v). We also sample one out of d node values, given sample rate $d > 0$, on paths of at least d nodes without any samples. The sampled values are stored in the same order as the nodes, and their positions are marked in bit vector B ($B[sp_u] = 1$, if node u is sampled).

Node values are retrieved in a similar way as in CSAs [19]. To retrieve $id(u)$, we first check if $B[sp_u] = 1$, and return sample $rank_1(B, sp_u)$, if this is the case. Otherwise we follow the only outgoing edge (u,v) by using function Ψ, and continue from node v. When we find a sampled node w, we return $id(w) - k$, where k is the number of steps taken by using Ψ.

4.3 Analysis

For each node $v \in V$, the length of range $[sp_v, ep_v]$ is the maximum of $in(v)$ and $out(v)$. As every node must have at least one incoming edge and one outgoing edge (except for the initial and the final nodes), the length of BWT is at most $2|E| - |V| + 2$.

Theorem 1. *Assume that rank and select on bit vectors require $O(t_B)$ time. GCSA with sample rate d supports* find*(P) in $O(|P| \cdot t_B)$ and* locate*($[sp_v, ep_v]$) in $O(d \cdot t_B)$ time.*

Proof. We use bit vectors Ψ_c that mark the occurrences of character $c \in \Sigma \cup \{\#\}$ to encode BWT. This reduces *rank* and *select* on BWT to the same operations on bit vectors. Basic operations ℓ and LF take $O(t_B)$ time, as they require a constant number of bit vector operations. Ψ also takes $O(t_B)$ time, if the current node has outdegree 1. As *find* does one generalized LF per character of pattern, it takes $O(|P| \cdot t_B)$ time.

Operation *locate* checks from bit vector B if the current position is sampled, and follows the unique outgoing edge using Ψ if not. This requires a constant number of bit vector operations per step. As a sample is found within $d-1$ steps, the time complexity is $O(d \cdot t_B)$. $\square$

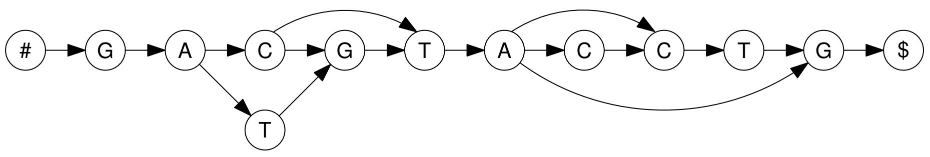


Fig. 3. A reverse deterministic automaton corresponding to the first 10 positions of the multiple alignment in Figure 1

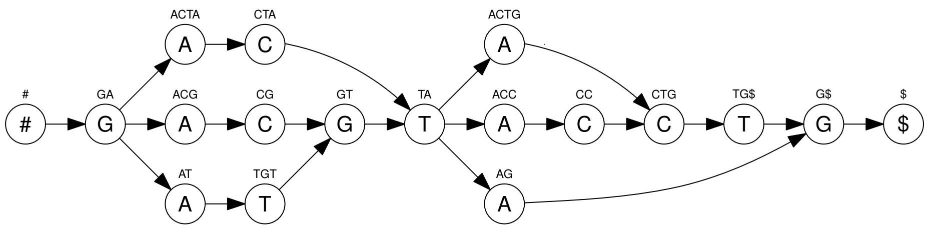


Fig. 4. A prefix-sorted automaton built for the automaton in Figure 3. The strings above nodes are prefixes $p(v)$.

Table 1. GCSA for the automaton in Figure 4. Nodes are identified by prefixes $p(v)$.

	`$`	`ACC`	`ACG`	`ACTA`	`ACTG`	`AG`	`AT`	`CC`	`CG`	`CTA`	`CTG`	`G$`	`GA`	`GT`	`TA`	`TG$`	`TGT`	`#`
BWT	`G`	`T`	`G`	`G`	`T`	`T`	`G`	`A`	`A`	`A`	`AC`	`AT`	`#--`	`CT`	`CG-`	`C`	`A`	`$`
F	1	1	1	1	1	1	1	1	1	1	10	10	100	10	100	1	1	1
M	1	1	1	1	1	1	1	1	1	1	10	10	111	10	111	1	1	1

4.4 Index Construction

Our construction algorithm is related to the *prefix-doubling* approach to suffix array construction [20]. We start with a reverse deterministic automaton (see Figure 3), convert it to an equivalent prefix-sorted automaton (Figure 4), and build the GCSA (Table 1) for that automaton.

Definition 2. *Let A be a finite automaton recognizing a finite language, and let $k > 0$ be an integer. A is* k-sorted *if, for every node $v \in V$, the labels of all paths from v to $v_{|V|}$ share a common prefix $p(v, k)$ of length k, or if node v is prefix-sorted by prefix $p(v, k)$ of length at most k.*

Starting from a reverse deterministic automaton $A = A_0$, we build a series of automata $A_i = (V_i, E_i)$ for $i = 1, 2, \ldots$ that are 2^i-sorted, until we get an automaton that is prefix-sorted. The construction algorithm is described and analyzed in more detail in the full paper[2]

[2] arXiv:1010.2656 [cs.DS].

Due to the lack of space, we just summarize the result of the analysis here. Assume that we have a random reference sequence of length n from an alphabet of size σ, with a random SNP at each position with probability $p < 0.25$. Then there are $f(n,p) = n(1+p)^{O(\log_\sigma n)} + O(1)$ nodes and edges in the expected case, and the algorithm uses $O(f(n,p) \log f(n,p) \log n)$ time and $O(f(n,p) \log f(n,p))$ bits of space. As we mostly use sorting, scanning and database joins, the algorithm can be implemented efficiently in parallel, distributed, and external memory settings.

5 Implementation and Experiments

We have implemented GCSA in C++, using the components from on our implementation of RLCSA [15][3]. For each character $c \in \Sigma \cup \{\#\}$, we use a gap encoded bit vector to mark the occurrences of c in BWT. Bit vectors F and M are run-length encoded. Bit vector B is gap encoded, while the samples are stored using $\lceil \log(id_{\max}+1) \rceil$ bits each, where $id_{\max}$ is the largest sampled value.

The implementation was compiled on g++ version 4.3.3. Index construction was done on a system with 128 gigabytes of memory and four quad-core Intel Xeon X7350 processors running at 2.93 GHz, while the other experiments were done on another system with 32 gigabytes of memory and two quad-core Intel Xeon E5540 processors running at 2.53 GHz. We used only one core in all experiments. Both systems were running Ubuntu 10.04 with Linux kernel 2.6.32.

We built a multiple alignment for four different assemblies of the human chromosome 18 (about 76 Mbp each). Three of the assemblies were from NCBI[4]: the assemblies by the Genome Reference Consortium (*GRCh37*), Celera Genomics, and J. Craig Venter Institute. The fourth sequence[5] was from Beijing Genomics Institute. The sequences were aligned by the Mauve Multiple Genome Alignment software [3]. We ran progressiveMauve 2.3.1 assuming collinear genomes, as we do not support rearrangements. The multiple alignment took a few hours to build: about 89.4% of nucleotides aligned perfectly, 0.19% with one or more mismatches, and 10.4% were inside of a gap. The number of gaps was high mainly because of the differences in the centromere region.

We constructed a GCSA (sample rate 16) for the alignment with various context lengths m. For each base $S_j[i]$ of sequence S_j, we used the next m bases as a context. We created a node for each base of each sequence, with an edge to the next base in the sequence. The nodes for aligned bases $S_j[i]$ and $S_{j'}[i']$ were then merged, if the bases and their contexts were identical. We also built RLCSA (sample rate 32) and BWA 0.5.8a [12] for the four sequences.

We searched for exact matches of 10 million Illumina/Solexa reads of length 56, sequenced from the whole genome, as both regular patterns and reverse complements. Table 2 lists results of these experiments. GCSA was 2.5–4 times slower than RLCSA and 3.5–5 times slower than BWA. About 1% of the reads matched

[3] `http://www.cs.helsinki.fi/group/suds/gcsa/`
[4] `ftp://ftp.ncbi.nih.gov/genomes/H_sapiens/Assembled_chromosomes/`
[5] `ftp://public.genomics.org.cn/BGI/yanhuang/fa/`

Table 2. Index construction and exact matching with GCSA (sample rate 16), RLCSA (sample rate 32), and BWA on a multiple alignment of four sequences of human chromosome 18. Times for *locate* include the time used by *find*. GCSA-m denotes GCSA with context length m.

		Construction		**Matching**		
Index	**Size**	**Time**	**Space**	**Matches**	**Find**	**Locate**
GCSA-2	71.8 MB	342 min	77 GB	389,292	16 min	25 min
GCSA-4	66.9 MB	295 min	29 GB	388,521	16 min	18 min
GCSA-8	64.5 MB	286 min	22 GB	387,807	16 min	17 min
RLCSA	165.0 MB	11 min	2.3 GB	384,400	6 min	7 min
BWA	212.4 MB	4 min	1.4 GB	384,400	-	5 min

by GCSA were not matched by the other indexes. Construction requirements for GCSA were high (but see Sect. 6 for discussion).

The performance gap between GCSA and RLCSA reflects differences in fundamental techniques, as the implementations share most of their basic components and design choices. Theoretically GCSA should be about 3 times slower, as it requires six bit vector operations per base in *find*, while RLCSA uses just two. The differences between RLCSA and BWA come from implementation choices, as RLCSA is intended for highly repetitive sequences and BWA for fast pattern matching with DNA sequences.

To test GCSA in a more realistic alignment algorithm, we implemented BWA-like approximate searching [12] for both GCSA and RLCSA. There are some differences to BWA: i) we return all best matches; ii) we do not use a seed sequence; iii) we have no limits on gaps; and iv) we have to match $O(|P| \log |P|)$ instead of $O(|P|)$ characters to build the lower bound array $D(\cdot)$ for pattern P, as we have not indexed the reverse sequence. We used context length 4 for GCSA, as it had the best trade-off in exact matching.

The results can be seen in Table 3. GCSA was about 2.5 times slower than RLCSA, while finding from 1.1% (exact matching) to 2.5% (edit distance 3) more matches in addition to those found by RLCSA. BWA is significantly faster (e.g. finding 1,109,668 matches with $k = 3$ in 40 minutes), as it solves a slighly different problem, ignoring a large part of the search space with biologically implausible edit operations. A fair comparison with BWA is currently impossible without significant amount of reverse engineering. With the same algorithm in all three indexes, the performance differences should be similar as in exact matching.

Finally, we made a preliminary experiment on the SNP calling application mentioned in Sect. 1 using in-house software. We called for SNPs from chromosome 18 with minimum coverage 2, using all 10 million reads, as well as only those reads with *no* exact matches on GCSA-4. The number of called SNPs was 4203 with all reads and 1074 with non-matching ones. We did not yet compare how much of the reduction can be explained by exact matches on recombinants that would also be found using approximate search on one reference and how much by more accurate alignment due to richer reference set.

Table 3. Approximate matching with GCSA and RLCSA. The reported matches for given edit distance k include those found with smaller edit distances.

	GCSA-4		**RLCSA**	
k	**Matches**	**Time**	**Matches**	**Time**
0	388,521	18 min	384,400	7 min
1	620,482	103 min	609,320	39 min
2	876,877	256 min	856,373	101 min
3	1,147,404	1,751 min	1,118,719	534 min

6 Discussion

Based on our experiments, GCSA is 2.5–4 times slower than a similar implementation of CSA used in the same algorithm. With typical mutation rates, the index is also not much larger than a CSA built just for the reference sequence. Hence GCSA does not require significantly more resources than a regular compressed suffix array, while providing biologically relevant extended functionality.

While our current construction algorithm uses much resources, recent developments have improved it significantly. The next implementation should be faster and use several times less memory than the current one. A parallel external memory implementation should allow us to build an index for the human genome and all known SNPs in a few days. Extrapolating from current results, the final index should be 2.5–3 gigabytes in size. We are also working on a different construction algorithm in the MapReduce framework [4].

To improve the running time of short read alignment and related tasks, most of the search space pruning mechanisms (in addition to the one mechanism we already used from [12]) to support approximate matching on top of BWT [10,12,13,16] can be easily plugged in.

As mentioned in Section 1, an obvious generalization is to index labeled weighted graphs, where the weights correspond to probabilities for jumping from one sequence to another in the alignment. This does not increase space usage significantly, as the probabilities differ from 1.0 only in nodes with multiple outgoing edges. During the construction of the index, it is also easy to discard paths with small probabilities, given a threshold. This approach can be used e.g. to index recombinants only in the recombination hotspot areas [18].

The experiments conducted here aimed at demonstrating the feasibility and potential of the approach. Once we have the genome-scale implementation ready, we are able to test our claims on improving variant calling and primer design accuracy with the index. Both require wet-lab verification to see the true effect on reducing false positives.

Acknowledgements. We wish to thank Eric Rivals for pointing out recombination hotspots and Riku Katainen for running the variation calling experiment.

References

1. Albers, C.A., et al.: Dindel: Accurate indel calls from short-read data. Genome Research (October 2010)
2. Burrows, M., Wheeler, D.: A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation (1994)
3. Darling, A.E., et al.: ProgressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement. PLoS ONE 5(6), e11147 (2010)
4. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: Proc. OSDI 2004, pp. 137–150. USENIX Association (2004)
5. Ferragina, P., et al.: Compressing and indexing labeled trees, with applications. Journal of the ACM 57(1), article 4 (2009)
6. Ferragina, P., Manzini, G.: Indexing compressed text. Journal of the ACM 52(4), 552–581 (2005)
7. Flicek, P., Birney, E.: Sense from sequence reads: methods for alignment and assembly. Nature Methods 6, S6–S12 (2009)
8. Grossi, R., Vitter, J.S.: Compressed suffix arrays and suffix trees with applications to text indexing and string matching. SIAM Journal on Computing 35(2), 378–407 (2005)
9. Lander, E.S., et al.: Initial sequencing and analysis of the human genome. Nature 409(6822), 860–921 (2001)
10. Langmead, B., et al.: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10(3), R25 (2009)
11. Levy, S., et al.: The diploid genome sequence of an individual human. PLoS Biol. 5(10), e254 (2007)
12. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 27(14), 1754–1760 (2009)
13. Li, R., et al.: SOAP2. Bioinformatics 25(15), 1966–1967 (2009)
14. Li, R., et al.: De novo assembly of human genomes with massively parallel short read sequencing. Genome Res. 20(2), 265–272 (2010)
15. Mäkinen, V., et al.: Storage and retrieval of highly repetitive sequence collections. Journal of Computational Biology 17(3), 281–308 (2010)
16. Mäkinen, V., et al.: Unified view of backward backtracking in short read mapping. In: Elomaa, T., Mannila, H., Orponen, P. (eds.) Ukkonen Festschrift 2010. LNCS, vol. 6060, pp. 182–195. Springer, Heidelberg (2010)
17. Metzker, M.L.: Sequencing technologies – the next generation. Nature Reviews Genetics 11, 31–46 (2010)
18. Myers, S., et al.: A fine-scale map of recombination rates and hotspots across the human genome. Science 310(5746), 321–324 (2005)
19. Navarro, G., Mäkinen, V.: Compressed full-text indexes. ACM Computing Surveys 39(1), 2 (2007)
20. Puglisi, S.J., et al.: A taxonomy of suffix array construction algorithms. ACM Computing Surveys 39(2), 4 (2007)
21. Spang, R., et al.: A novel approach to remote homology detection: Jumping alignments. Journal of Computational Biology 9(5), 747–760 (2002)
22. Venter, J.C., et al.: The sequence of the human genome. Science 291(5507), 1304–1351 (2001)
23. Wheeler, D.A., et al.: The complete genome of an individual by massively parallel DNA sequencing. Nature 452(7189), 872–876 (2008)

Efficiently Solvable Perfect Phylogeny Problems on Binary and *k*-State Data with Missing Values

Kristian Stevens[1] and Bonnie Kirkpatrick[2]

[1] Computer Science and Evolution and Ecology, University of California Davis
`kastevens@ucdavis.edu`

[2] Electrical Engineering and Computer Sciences, University of California Berkeley
`bbkirk@eecs.berkeley.edu`

Abstract. The perfect phylogeny problem is of central importance to both evolutionary biology and population genetics. Missing values are a common occurrence in both sequence and genotype data. In their presence, the problem of finding a perfect phylogeny is NP-hard, even for binary characters [24]. We extend the utility of the perfect phylogeny by introducing new efficient algorithms for broad classes of binary and multi-state data with missing values.

Specifically, we address the *rich data hypothesis* introduced by Halperin and Karp [11] for the binary perfect phylogeny problem with missing data. We give an efficient algorithm for enumerating phylogenies compatible with characters satisfying the *rich data hypothesis*. This algorithm is useful for computing the probability of data with missing values under the coalescent model.

In addition, we use the *partition intersection* (PI) graph and chordal graph theory to generalize the *rich data hypothesis* to multi-state characters with missing values. For a bounded number of states, k, we provide a fixed parameter tractable algorithm for the k-state perfect phylogeny problem with missing data. Our approach reduces missing data problems to problems on complete data. Finally, we characterize a commonly observed condition, an m-clique in the PI graph, under which a perfect phylogeny can be found efficiently for binary characters with missing values. We evaluate our results with extensive empirical analysis using two biologically motivated generative models of character data.

1 Introduction and Background

A current and central problem in the study of molecular evolution concerns the reconstruction of evolutionary history in the form of *phylogenetic trees*. Extant haploid genomes (*taxa*) correspond to the tree leaves, while internal vertices correspond to hypothetical ancestral taxa. Mutation is the primary process on these trees, and each branch represents a historical opportunity for mutations. The *perfect* phylogeny is a phylogeny that permits only one mutation at each genomic site (*character*). This means that each edge in the tree is labeled with the character(s) that mutate, and the vertices of the tree are labeled with haplotypes, sequences of character states.

Perfect phylogenies have been used to obtain the haplotypes of diploid individuals from genotype information [25]. Indeed, recent methods of whole genome sequencing are able to find polymorphic sites at an unprecedented resolution where recombination is unlikely to occur. The haplotypes inferred from genotype data are used to estimate recombination rates [16] and to do case-control association mapping to find disease loci [4]. Furthermore, general phylogenies have been used to model somatic genome evolution and understand cancer genomes [21]. The perfect phylogeny may be ideally suited to studying cancer genomes, since somatic genomes undergo mitosis without recombination.

The perfect phylogeny has a very close relationship to the *coalescent* of population genetics. A perfect phylogeny on binary characters results under the commonly used *infinite-sites* model of evolution, characters are binary and only mutate once in the history of the sample. A perfect phylogeny where characters have more than two states occurs under the *infinite-alleles* model.

Formally, a *phylogenetic tree* $\mathcal{T}$ is an ordered pair (T, Φ). Here $T = (V, E)$ is a tree, and the *labeling map* $\Phi : X \rightarrow \mathcal{P}(V)$, where X is a set of *taxa* with cardinality n, and $\mathcal{P}(V)$ is the powerset of V. Under the perfect phylogeny model, a taxon always labels some connected subtree [11]. A *fully labeled phylogenetic tree* is a phylogenetic tree where each taxon maps to a single vertex.

A *full character* is a map $C : X \rightarrow A$ from the taxa to the set of character states A with cardinality k. If $k = 2$, the character is called a binary character and the states are labeled $A = \{0, 1\}$. A non-full character, $\mathcal{C}$, is a *partial character*, and can be written as $C : X \rightarrow A \cup \{*\}$ where the star indicates unspecified states from missing data.

Let a *collection of partial characters* be $\mathcal{C} = \{C_1, ..., C_m\}$. Let X be the union of the domains of $C \in \mathcal{C}$. We say that a character C for taxon $x \in X$ has a *specified* state if $C(x) \neq *$. We do not consider non-informative partial characters with less than two states in A, i.e. characters C where $|C(X)| < 2$. A *resolution* of $\mathcal{C}$ is a full character set $\mathcal{C}^{\mathcal{R}}$ which agrees with $\mathcal{C}$ in every specified character state and gives an assignment for the unspecified states.

A character $C : X \rightarrow A \cup \{*\}$ is *convex* on a phylogenetic tree $\mathcal{T}$ if and only if there is an extension $\bar{C} : V \rightarrow A$ of the character such that the following conditions hold: i) The restriction of the domain of $\bar{C}$ to the specified states yields a function that agrees with C. ii) Let T_a be the subgraph of T induced by the set of vertices mapped by $\bar{C}$ to a particular character state a. For every character state $a \in A$, T_a is connected and disjoint from all other T_b for $b \in A$.

We say that a set of full characters is *compatible* if and only if there is a phylogenetic tree on which all the characters are convex. A set of partial characters $\mathcal{C}$ is *compatible* when there is a fully-specified resolution $\mathcal{C}^{\mathcal{R}}$ that is compatible.

Consider a set of partial characters $\mathcal{C} = \{C_1, ..., C_m\}$ that are compatible with a fully labeled phylogenetic tree $\mathcal{T} = (T, \phi)$. For character C_i, let $\bar{C}_i$ be the extension of C_i on tree T. For a vertex $v \in V$ in the tree, let the *haplotype* $h(v)$ be a sequence of character states, one for each character, where the $h_i(v) = a$ if and only if $v \in T_a$ for character $\bar{C}_i$. Then each edge (u, v) of the tree can be labeled with the characters whose states change between $h(u)$ and $h(v)$. For a fully labeled phylogenetic tree, the labeling function ϕ yields a haplotype for each taxon x, $h(\phi(x))$.

The *perfect phylogeny problem* is as follows: given a set of characters, determine whether the characters are compatible and construct a tree on which the characters are convex.

For full binary characters, there are $O(nm)$ algorithms for the perfect phylogeny problem, e.g. [28]. For case of $k = 3$ an $O(nm^2)$ algorithm was first shown in [6]. When the full characters can take on at most k states, the perfect phylogeny problem was found to be fixed parameter tractable in [1]. An improved approach for bounded k is that of Kannan and Warnow [13] which has a running time of $O(2^{2k}nm^2)$.

On partial characters, the perfect phylogeny problem is known to be NP-hard [24] even in the binary case. A common restricted instance, perfect phylogeny on directed binary partial characters, is solvable in polynomial time when the state transitions are all *directed* from $0 \rightarrow 1$. This can be achieved under the perfect phylogeny model if any complete taxon is available in the matrix. The fastest solution to the directed problem is the algorithm of Pe'er et, al. which runs in almost linear time [20]. Algorithms for solving the general problem on binary partial characters are described in [10] using integer linear programming and [22] by enumerating potential roots for the directed problem after an effective missing data reduction phase.

The Rich Data Hypothesis. For binary characters C_i and C_j, let $V(C_i, C_j) \subseteq \{(0,0),(0,1),(1,0),(1,1)\}$ be the set of values that the pair of characters takes over the the observed taxa in $\mathcal{C}$. The classic *splits-equivalence theorem* [23], also known as the four-gamete condition, states that a collection of full binary characters is *compatible* if and only if $|V(C_i, C_j)| \leq 3$ for all pairs of characters C_i and C_j.

Definition 1 (rich data hypothesis). *A set of binary partial characters where* $|V(C_i, C_j)| = 3$ *for all pairs of characters* C_i *and* C_j *satisfies the* rich data hypothesis (RDH) *of Halperin and Karp [11].*

The rich data hypothesis is of theoretical importance in that it characterizes a class of tractable perfect phylogeny problems on binary partial characters. Halperin and Karp demonstrated that under a biologically motivated model, described later, data will frequently conform to the rich data hypothesis. The rich data hypothesis does not restrict the topology of trees in any way. And clearly, not every collection of binary partial characters satisfying the RDH definition is compatible. One drawback of RDH is that it does not allow for more than one mutation per tree edge. This restriction is exploited by Halperin and Karp to achieve a near-linear time perfect phylogeny algorithm for RDH characters.

We describe an efficient algorithm for finding *all* the perfect phylogeny trees compatible with binary partial characters that satisfy the rich data hypothesis in section 2. Our $O(nm^2)$ algorithm extends the tree-popping algorithm of Meacham [19]. Using tree-popping, we are able to inductively prove that the tree topology T for RDH characters is unique. The enumeration algorithm can be used to compute the likelihood of RDH characters under the coalescent model or for Bayesian inference. This allows for estimation of the coalescent genealogy and mutation rates in the presence of missing data.

Since the RDH only applies to binary partial characters where no two characters label the same tree edge, we introduce more general criteria than the rich data hypothesis that also allow for efficient solutions to the perfect phylogeny problem with missing data.

These new criteria are motivated by the RDH and leverage chordal graph theory and the partition intersection graph to dramatically increase the frequency with which our simulations found a perfect phylogeny in the presence of missing data. To generalize the RDH we utilize a reduction from the missing data problem to a problem on complete data. We give an $O(2^{2k}nm^2)$ algorithm for inferring a perfect phylogeny from k-state partial characters satisfying a generalized interpretation of the RDH. We also give an $O(nm^2)$ algorithm for directing binary partial characters for a large class of problem instances which includes the RDH.

We conclude with an extensive empirical analysis under two biologically motivated generative models: the coalescent and the finite haplotype model of Halperin and Karp [11]. The results show a dramatic improvement over previously characterized methods. This is particularly true under the highly relevant coalescent model.

2 Enumerating Fully Labeled Phylogenetic Trees

In this section we introduce the tree-popping enumeration algorithm for binary partial characters satisfying the RDH. Enumeration allows computation of the likelihood of the data, or the probability of the data being generated by the coalescent model with infinite sites. Genealogies from the coalescent with infinite sites are always consistent with the perfect phylogeny tree [29]. The perfect phylogeny models only the tree structure, while the coalescent genealogy also models the temporal order of coalescent and mutation events. For a given set of binary partial characters, there may be multiple perfect phylogenies consistent with the data. Thus, computing probabilities of the data requires integrating over the perfect phylogenies consistent with the data, and for each perfect phylogeny, computing the probability of the data over all coalescent genealogies consistent with that perfect phylogeny.

When given m partial characters on n taxa that satisfies RDH, we can modify Meacham's tree-popping algorithm for full characters [19] to compactly represent all the fully labeled phylogenetic trees. A polynomial-time version of tree-popping was reviewed in detail in [23]. Details for a linear-time tree-popping algorithm on full characters are given in [28].

Our TREE-POPPING algorithm for RDH characters runs in $O(nm^2)$ time and works by iteratively modifying a tree to respect the bipartition on the taxa that is described by character C_i. The tree begins as a single vertex labeled with all the taxa. For each new character, we find the unique vertex, b, that is labeled with taxa having both character states $0, 1$ of C_i. A simple version of this step takes $O(nm)$ time. We then add a single edge to the tree by splitting b into two vertices b_0 and b_1, each labeled with the taxa of b that belong to a single bipartition of C_i. The edges incident to b, are then connected to either b_0 and b_1 in a manner preserving convexity, and the taxa labels Φ are updated to be consistent with the convexity of character C_i. This splitting operation is called *tree-popping.*

Using our TREE-POPPING algorithm, we prove that the tree topology for compatible RDH characters is unique, implying that the enumerated trees only differ in the labeling of the taxa on the vertices. The tree-popping algorithm represents all the fully labeled phylogenetic trees by representing, for each taxon, the subgraph of the phylogenetic tree that a taxon can compatibly label. At each step of the tree-popping, there is there is a unique vertex b that is labeled with both states of the new character C_i, the tree-popping algorithm tree-pops this vertex generating a unique change to the tree between the $i-1$ and ith steps. Since the beginning tree is unique, each step is determined, and the resulting tree is invariant to permutation of the first i characters, the tree topology is unique. Since the topology is unique, the set of haplotypes that label the vertices of the tree are also well-defined and unique. Since a taxon may be consistent with multiple tree haplotypes, there can, however, be multiple resolutions of of a particular missing data value. This results in multiple fully-labeled phylogenetic trees.

Indeed, the possible fully labeled phylogenetic trees differ only in the labeling functions $\phi : X \to V$ which map each taxon to a single vertex. In general, phylogenetic tree $\mathcal{T} = (T, \Phi)$ can be fully labeled using $\prod_{s \in X} |\Phi(s)|$ different ϕ. So, for partial characters $\mathcal{C}$ the probability of observing them under the coalescent model is $\mathbb{P}[\mathcal{C}] = \sum_{\phi} \mathbb{P}[\mathcal{C}|\phi, T]\mathbb{P}[\phi|T]\mathbb{P}[T]$ where $\mathbb{P}[\phi|T]$ is uniform over ϕ and where $\mathbb{P}[\mathcal{C}|\phi, T]$ is the probability of the data on coalescent genealogies constrained by ϕ and T. Since the topology is unique under the RDH, $\mathbb{P}[T]$ is one. An effective method for computing the coalescent probability is given in [29]. Integration over the possible trees can be done exhaustively or by Monte-Carlo sampling of ϕ. $\mathbb{P}[\mathcal{C}]$ is the likelihood and can be used for maximum likelihood or Bayesian parameter inference.

3 Generalizing the Rich Data Hypothesis

In this section we characterize sufficient conditions and polynomial-time algorithms for perfect phylogeny on partial characters, and generalize the *rich data hypothesis.* The approach taken here relies heavily on chordal graph theory [8] [17] and the partition intersection graph [3] [17] [23].

In the *partition intersection (PI)* graph for a set of characters $\mathcal{C}$, denoted by $int(\mathcal{C})$, each character in $\mathcal{C}$, $C : X \to A$, induces a partition on the taxa given by $C^{-1}(A)$. The PI graph $int(\mathcal{C}) = \{V_I, E_I\}$ summarizes the co-occurrence of character, state pairs on the observed taxa. The PI graph $int(\mathcal{C})$ is defined on $\mathcal{C}$ as follows: there is a vertex in V_I for each character, state pair. For a binary character C_i there will be vertices corresponding to C_i^0 and C_i^1. There is an edge between two vertices when their character state pairs co-occur in some taxon s in X. In other words $(C_i^a, C_j^b) \in E_I$ if there is some taxon s such that $C_i(s) = a$ and $C_j(s) = b$. The PI graph for a set of full characters is m-partite and m-colorable with colors corresponding to characters [17] [18]. A *legal* edge in the PI graph must connect two vertices belonging to different colors or characters.

An edge between two nonconsecutive vertices in a cycle is referred to as a *chord* of that cycle. We note that a graph G is said to be *chordal* if every cycle in G with greater than three vertices has a chord connecting non-adjacent vertices in the cycle. So, a graph is *chordal* when there are no chord-less cycles. A *triangulation* of a graph G is a chordal graph on the same vertex set but with additional edges making it chordal. A *legal triangulation* $H(int(\mathcal{C}))$, of the PI graph $int(\mathcal{C})$, is a triangulation where all the edges are *legal*.

The following classic theorem, originally due to Buneman, is valid for partial characters with an arbitrary number of states:

Theorem 1. *The collection of characters $\mathcal{C}$ is compatible if and only if there exists a legal triangulation, $H(int(\mathcal{C}))$, of $int(\mathcal{C})$. [3] [23]*

We now show that a legally triangulated partition intersection graph has a perfect phylogeny. Given any legal triangulation of $int(\mathcal{C})$, it is possible to derive a fully labeled perfect phylogeny tree. Let $G = (V, E)$ be an arbitrary undirected graph. A *clique tree* is defined as a tree $R(G) = (W, E)$ having one vertex $w \in W$ for each max-clique $\gamma(w)$ in G and satisfying the intersection property. The *intersection property* states that there is a connected subtree of R induced by graph vertex $v \in V$, i.e. the subgraph induced by tree vertices $\{w \in W : v \in \gamma(w)\}$. A graph G is chordal if and only if it has a clique tree $R(G)$ [2][8][17]. Therefore, from a legal triangulation of $int(\mathcal{C})$, one can build a clique tree, $R(H(int(\mathcal{C})))$. Clearly $R(H(int(\mathcal{C})))$ is a perfect phylogeny on which $\mathcal{C}$ are convex, because each vertex in $int(\mathcal{C})$ corresponds to a particular character state and the intersection property guarantees that each character state is convex. This means that each $C \in \mathcal{C}$ has at least one extension $\bar{C}$ defined by the max-cliques of $H(int(\mathcal{C}))$ in which C participates and that $\bar{C}$ is convex on $R(H(int(\mathcal{C})))$.

It is convenient to extend our notion of haplotypes to the cliques of $int(\mathcal{C})$. Recall that each vertex, v of a perfect phylogeny has a haplotype $h(v)$. Now, consider a vertex w of the clique tree. This vertex corresponds to a max-clique $\gamma(w)$. If $\gamma(w)$ is an m-clique, then it must contain one vertex from each character. Therefore, given clique $\gamma(w)$, we simply read off the haplotype $h(\gamma(w))$ which is the character states appearing in the clique for each of the characters in order.

The reduction from the problem of finding a perfect phylogeny on partial characters $\mathcal{C}$ to the problem of finding finding a legal triangulation of $int(\mathcal{C})$ implies that the latter problem is NP-hard even for binary characters. An effective approach to the legal triangulation problem, for data of moderate size, using integer linear programming, was recently described in [9].

In contrast, once the PI graph has been legally triangulated, it is known that a clique tree can be constructed in time linear in the size of the PI graph [2]. This will give us another way of observing why the RDH permits efficient algorithms:

Lemma 1. *For a set of binary partial characters that satisfies the RDH, $int(\mathcal{C})$ is legally triangulated if and only if $\mathcal{C}$ has a perfect phylogeny.*

Lemma 1 can be easily shown by the fact that adding *any* legal edge to $int(\mathcal{C})$ creates a four vertex cycle containing two colors that can only be triangulated by adding an additional edge connecting two vertices of the same color. It suggests a way to generalize the RDH. It also implies an efficient alternative solution to perfect phylogeny problem for characters that satisfy the RDH: use a linear algorithm for recognizing a chordal graph [27], build a clique-tree from the graph [2], and construct the corresponding perfect phylogeny [9]. We call this the *chordal* method and examine its performance in detail later. The *chordal* method also efficiently builds a perfect phylogeny from k-state characters, but only if their PI graph is already chordal. Halperin and Karp presented their own algorithm, analogous to finding a tree from a PI graph on binary RDH characters. They reduce the problem to 2-SAT. Since their algorithm depends on $|V(C_i, C_j)| = 3$, it is less general than the *chordal* method.

3.1 The Generalized Rich Data Hypothesis

In this section we build on previous observations to generalize the rich data hypothesis to an arbitrarily bounded number of states k, and provide conceptual solutions to missing data problems when the PI graph is *not* chordal. The main idea is to reduce missing data problems to complete data problems. We will first introduce the following definition:

Definition 2 (GRDH). *A set of partial characters, $\mathcal{C}$, satisfy the generalized rich data hypothesis (GRDH) when there exists a resolution of the characters $\mathcal{C}^{\mathcal{R}}$ such that $int(\mathcal{C}) = int(\mathcal{C}^{\mathcal{R}})$.*

Notice this definition is a property only of the partial characters and does not mention compatibility of the partial characters or their resolution. The GRDH does not imply the PI graph is chordal. The definition includes all binary RDH data for which there is a perfect phylogeny. Unlike the RDH, under the GRDH two mutational events (characters) can label the same tree edge. This definition also clearly applies to multi-state data. Finally, the feature that makes this definition particularly useful for multi-state data is, as we will show, that it defines a set of missing data problems on k-state data that can be efficiently solved for bounded k in time polynomial in n, and m. It is also straightforward to establish that Halperin and Karp's argument that the RDH holds for a set of characters with high probability for sufficiently large n under their biologically motivated probibalistic generative model also applies to the GRDH.

Our approach uses an equivalent characterization of the GRDH. We say a clique *covers* the edge between vertices u and v if both u and v are in the clique's vertex set. An *edge clique cover* for a graph G is a set of cliques in G covering all the edges of G. The PI graph for any set of full characters will have an a edge clique-cover consisting of max-cliques with m vertices, which we refer to as an m-clique cover [17][18]. Importantly, if $int(\mathcal{C})$ satisfies the GRDH, then it must also have an m-clique cover, since resolving the characters does not add edges to the PI graph $int(\mathcal{C}^{\mathcal{R}})$.

We first establish that a problem on partial characters $\mathcal{C}$, can be reduced to a problem on complete characters $\mathcal{C}^{\mathcal{R}}$ given an m-clique cover for $int(\mathcal{C})$. In the next section we show a simple efficient algorithm, FIND-m-CLIQUES, for determining if $int(\mathcal{C})$ has an m-clique cover, by enumerating all distinct m-cliques of $int(\mathcal{C})$ up to a bound imposed by the maximum number of m-cliques possible in a legal triangulation of $int(\mathcal{C})$.

Lemma 2. *The PI graph of any set of full characters $\mathcal{C}^{\mathcal{R}}$ must have an m-clique cover, where an m-clique cover is a collection of max-cliques covering the edges of $int(\mathcal{C})$ with each clique having m vertices [17][18].*

Lemma 3. *Let $\mathcal{C}$ be a set of partial characters with PI graph $int(\mathcal{C})$. Any m-clique in $int(\mathcal{C})$ specifies a haplotype that will appear as a vertex in* every *perfect phylogeny, if the characters are compatible.*

Proof. An m-clique in int(C) appears as a maximal clique in every legal triangulation $H(int(\mathcal{C}))$, because an m-clique is always a maximum clique for an m-partite graph. Now we simply want to show that the m-clique appears in every perfect phylogeny. Assume that there is a perfect phylogeny without the m-clique. We can use the perfect phylogeny to find a legal triangulation of $int(\mathcal{C})$ using Buneman's first theorem [3]. This legal triangulation must not contain the clique corresponding to the m-clique, because the phylogeny does not. But since $H(int(\mathcal{C}))$ is a chordal supergraph of $int(\mathcal{C})$, the legal triangulation must contain the m-clique. And this contradiction proves the claim. □

Lemma 4. *A collection of GRDH partial characters $\mathcal{C}$ is compatible if and only if $int(\mathcal{C})$ has an m-clique cover and each m-clique appears as a haplotype in a perfect phylogeny.*

Proof. By the definition of the GRDH and lemma 2, $int(\mathcal{C})$ has an m-clique cover. Using the established correctness of Buneman's theorem for missing data problems, $\mathcal{C}$ has a perfect phylogeny if and only if $int(\mathcal{C})$ has a legal triangulation $H(int(\mathcal{C}))$. Also $H(int(\mathcal{C}))$ is chordal if and only if it has a clique-tree labeled by the maximal cliques of $H(int(\mathcal{C}))$. Furthermore, there exists a resolution $\mathcal{C}^{\mathcal{R}}$ obtained from $H(int(\mathcal{C}))$, since every clique in $H(int(\mathcal{C}))$ is a chordal supergraph of the cliques in $int(\mathcal{C})$. Suppose there is a clique-tree for $H(int(\mathcal{C}))$. Since both $int(\mathcal{C})$ and $H(int(\mathcal{C}))$ are m-partite, each m-clique in $int(\mathcal{C})$ is also an m-clique in $H(int(\mathcal{C}))$ and must be maximal. Therefore each m-clique in $int(\mathcal{C})$ must appear as a vertex in this clique tree. Using the argument that obtains a perfect phylogeny from clique tree, each m-clique appears as a haplotype in a perfect phylogeny. □

Lemma 3 establishes that every m-clique in $int(\mathcal{C})$ corresponds to a haplotype in any perfect phylogeny for $\mathcal{C}$. Lemma 2 implies that that if $\mathcal{C}$ satisfies the GRDH, it must also have m-clique cover. And finally, Lemma 4 establishes that the resolved characters $\mathcal{C}^{\mathcal{R}}$ defined by the m-clique cover can be substituted for the partial characters. This reduction only works when the partial characters $\mathcal{C}$ satisfy the GRDH. It remains to describe an approach for determining the m-clique cover.

3.2 Finding m-Cliques

We begin by computing the maximum number of m-cliques in a legally triangulated PI graph. This bound is essential to contain the complexity of the m-clique enumeration algorithm. The approach taken here is similar in structure to the chordal graph recognition algorithm of Fulkerson and Gross [7].

An important property of chordal graphs is the inheritance property [2]. It is the fact that when any number of vertices are removed from a chordal graph, it remains chordal (Lemma 5). A vertex s is *simplicial* if the subgraph induced by the vertices adjacent to s forms a clique (or simplex). An important property of chordal graphs with respect to simplicial vertices is stated as Lemma 6. It was characterized by Dirac [5] and Lekkerkerker and Boland [14]. A fact of simplicial vertices, proven in [15], is stated as Lemma 6.

Lemma 5. *Any induced subgraph of a chordal graph is also chordal. [2]*

Lemma 6. *A non-trivial chordal graph has at least two simplicial vertices. [5]*

Lemma 7. *A vertex is simplicial iff it is contained in exactly one maximal clique. [15]*

Lemmas 5, 6, and 7 are useful to determine the upper bound on the number of m-cliques for any m-partite chordal graph. Our approach iterates by counting a simplicial vertex which by Lemma 7 must belong to at most one m-clique since m-cliques are maximal. After counting the potential m-clique that contains the simplicial vertex, we remove it from the PI graph. We may then examine the other simplicial vertices of the same color, count potential m-cliques, and remove them. If all vertices of this color are removed, we stop counting and terminate. If a vertex is not simplicial then we can safely remove it from the PI graph without decreasing the number of m-cliques counted in the resulting subgraph. This easily proven fact is given here as Lemma 8.

Lemma 8. *If a non-simplicial vertex $\bar{s}$ is removed from G, the number of simplicial vertices in the subgraph $G/\{\bar{s}\}$ does not decrease.*

The counting procedure can be applied recursively with decreasing m. It is maximized when $k-1$ vertices are simplicial for all colors but the last. When there is only one color left there will be k m-cliques. Lemma 9 states an upper bound on the number of m-cliques in a chordal PI graph with k vertices per color. This is used by Theorem 2 to efficiently find all m-cliques.

Lemma 9. *The legally triangulated m-partite PI graph G on data with k states has at most $k + (k-1)(m-1)$ cliques of size m.*

Theorem 2. *[Find m-Cliques] If it is possible to legally triangulate a PI graph $int(\mathcal{C})$, all m-cliques in $int(\mathcal{C})$ can be found in $O(k^2m^3)$ time.*

Proof. Proceed by induction on m. For m=1 we remove an arbitrary character C from $\mathcal{C}$ and initialize a candidate list $\mathcal{Q}$ of all 1-cliques by adding a clique Q to $\mathcal{Q}$ corresponding to each state of C. We then induct on m. We remove a remaining character C from $\mathcal{C}$. All m cliques must contain an $m-1$ clique from the candidate list as a subgraph as well as one of the vertices colored by the current character C. For each state v of C and each clique Q in $\mathcal{Q}$ we can check if $Q \cup \{v\}$ is a clique. If it is, we place $Q \cup \{v\}$ in a new candidate list $\mathcal{Q}'$. Otherwise we discard it since it will not appear as a subgraph of any clique in the final list of $|\mathcal{C}|$-cliques. Checking if a vertex v can extend a clique in the candidate list can be done in $O(m)$ time. For the current C, there are at most k vertices to check. There are at most $O(km)$ cliques that are checked for extension. If we find more cliques than allowed by Lemma 9 at any time, we conclude that no perfect phylogeny exists. A round of extension is done in $O(k^2m^2)$ time. There are m characters in the induction, leading to an overall complexity of $O(k^2m^3)$. □

When there are $k > 3$ states in the data, we use the algorithm of Kannan and Warnow on $\mathcal{C}^{\mathcal{R}}$ obtained from the m-clique cover to find a perfect phylogeny for partial characters $\mathcal{C}$ satisfying the GRDH. For smaller k, the faster specialized algorithms may be applied. For k-state characters satisfying the GRDH, the time complexity of the entire procedure is $O(2^{2k}nm^2)$, assuming $n > m$.

3.3 Directed Binary Partial Characters

For binary partial characters it is possible to find a perfect phylogeny when the character state transitions are directed $0 \rightarrow 1$. The most efficient algorithm for directed characters is the near-linear time algorithm by Pe'er et, al. [20].

A surprising but obvious outcome of being able to efficiently enumerate *all* m-cliques of any PI graph which can be legally triangulated, is that the solution to the problem of finding a single m-clique is efficiently implemented by the same algorithm. This is remarkable, because the very similar problems of finding the clique of maximum size and an m-clique in an m-partite graph, such as the PI graph, are NP hard [26]. The reason we have an efficient algorithm is because we bound the enumeration using the total number of m-cliques possible in a legally triangulated PI graph. In Theorem 3, we state an efficiently computable, very general, sufficient condition on the data for directing binary characters:

Theorem 3. *[m-clique rooted] A set of binary partial characters, $\mathcal{C}$, can be directed in $O(nm^2)$ time when $int(\mathcal{C})$ contains an m-clique.*

Theorem 3 follows directly from Theorem 2 (for $k = 2$ and $n > m$) and the fact that every m-clique corresponds to a haplotype in every perfect phylogeny (Lemma 3). Theorem 3 applies to all binary data satisfying the *rich data hypothesis*, having an m-clique cover, or having a complete taxon. An important distinction between this characterization and the RDH is that the unlabeled tree is not always unique. In the next section we will demonstrate how commonly the m-clique condition is observed.

4 Empirical Results and Discussion

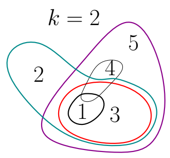
In this section, we evaluate the following implementations: *tree-popping*, *chordal*, *m-clique cover*, single *taxon rooted*, and single *m-clique rooted*. Fig. 1 lists these methods along with the algorithms that they implement, and, for binary characters, shows a cartoon Venn diagram of which data inputs these methods apply to. The similar Venn diagram (not shown) for $k > 2$ simply has two overlapping sets, for *chordal* and *m-clique cover*, where neither set includes the other completely.

Probabilistic Biological Generative Models. For our empirical analyses, we consider two biologically motivated models, the *coalescent* model and Halperin and Karp's *finite haplotype* model. The RDH was originally motivated by a model that assumes the population, from which individuals are sampled, consists of a *finite* number of haplotypes with unknown frequency. Under both models, the number of characters, m, states, k, sample size, n, and missing data rate, q, are specified, and a simulated data matrix is returned.

For the *coalescent* model we used Hudson's `ms` [12] to generate initial matrices of complete binary data under the the infinite-sites model with n sampled individuals. Each entry of the matrix was masked with probability q independently. The first m segregating characters were taken. To convert binary infinite-sites data, to data under an infinite alleles model (bounded by k) we applied the method described in [9] before masking.

For the *finite haplotype* model we begin with a fixed number of haplotypes consisting of *unique* characters (the original RDH requirement for $k = 2$). To obtain $h \leq k+(k-1)(m-1)$ unique haplotypes derived from a perfect phylogeny, we used our coalescent simulator without masking and take the first m *unique* characters. To obtain a sample of size n with missing data, the haplotypes were sampled with probability $1/h$ and each position was masked with probability q.

In the context of population genetics, an important distinguishing feature between the two models, is that under the coalescent, for a fixed number of randomly chosen characters, the frequency of the rarest haplotype in the population decreases as the sample size increases. This is because the coalescent models biological evolution without the ascertainment bias of polymorphism discovery.



	Name	Implementation	Time
1	*tree-popping (RDH)*	TREE-POPPING	$O(nm^2)$
2	*chordal*	We used [27] with [9]	$O(nm^2)$
3	*m-clique cover (GRDH)*	FIND m-CLIQUES + [13]	$O(2^{2k}nm^2)$
4	*taxon rooted*	We find haplotype + [20]	$\tilde{O}(nm)$
5	*m-clique rooted*	FIND m-CLIQUES + [20]	$O(knm^2)$

Fig. 1. (left) For $k = 2$, the cartoon Venn diagram shows the space of input data and each set illustrates the data instances for which a perfect phylogeny can be found with the numbered method. **(right)** The methods are numbered corresponding to the Venn diagram. Each method is described and is named as it is referred to in later plots.

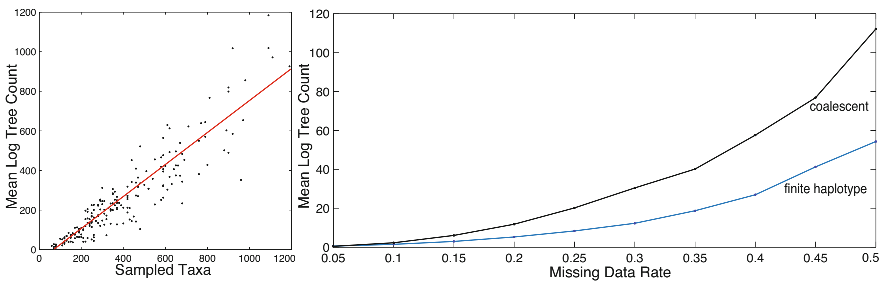


Fig. 2. The number of labeled trees grows as the missing data rate increases. **(right)** For $p = 0.5$, mean $log(l)$ is plotted against the number of taxa sampled under *coalescent* with $m = 10$ until the matrix was RDH for 100 trials. As expected, the regression line (red) shows a good linear fit (see text). We observe that the slope of this line increases with the missing data rate. **(left)** Mean $log(l)$ is plotted against p is super-linear.

The finite haplotype model is motivated by genotyping studies where ascertainment biases the minor allele frequencies.

Counting RDH Trees. For many applications, it is important to consider all trees consistent with the data. In Fig. 2 we characterize the explosive growth in the number of trees as the amount of missing data increases. Under the *coalescent* model, for a fixed $m = 10$, we increase n until a perfect phylogeny could be obtained with *tree-popping*. We then computed $log(l)$, where $l = \prod_{s \in X} |\Phi(s)|$ is the number of fully labeled phylogenetic trees. For a fixed missing data rate q, the observed relationship between $log(l)$ and n is approximately *linear*. Since l can be computed as the product over taxa of the number of vertices in the subtree each taxon can label, a log-linear relationship follows intuitively. The slope corresponds to the expected log of the size of the subtree over which a taxon is valid. The super-linear relationship of $log(l)$ to q, is explained by a simultaneous increase in the number of taxa required to obtain a perfect phylogeny and the expected size of the subtree over which a taxon is valid.

Generalizing the Rich Data Hypothesis. We investigated the performance of our algorithms under both models and compare to previous results. We examine *m-clique cover* first. In Fig. 3 we compare the relative performance of *m-clique cover* for data obtained under the *finite haplotype* model for k =2, 3, and 4. As intended, for fixed m, and large enough n, the matrix will satisfy the GRDH with high probability regardless of the number of states in the data. We compare the relative performance of *m-clique cover* to the original RDH criterion implemented with *tree-popping* in Fig. 3 under both models. For binary characters, we observed that data frequently satisfied GRDH but not the RDH. For example, a complete data matrix can be constructed with *m-clique cover* even if some haplotypes are missing due to under-sampling. Under the coalescent model, this difference is heightened due to a large number of mutation events observed on the same tree edge.

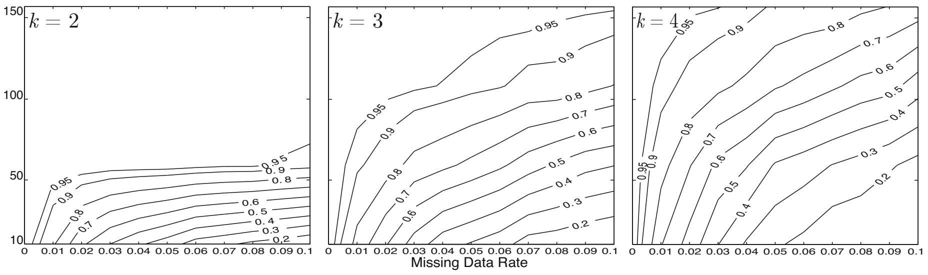


Fig. 3. Contour plots of the probability of an m-clique cover, for $m = 10$ and $k = 2$, 3, and 4. Each plot consists of 2×10^5 draws from the *finite haplotype* model. We observe empirically that with high probability the matrix will have an m-clique cover for sufficiently large n. As expected from theory, as k increases the number of samples required also increases due to the larger number of m-cliques that need to be observed in the data.

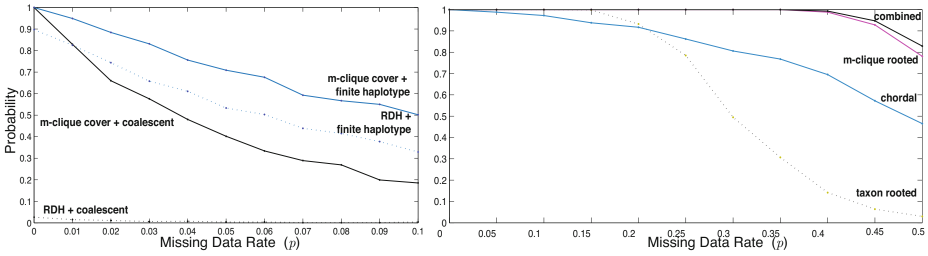


Fig. 4. A thousand simulations were performed for each missing data rate plotted for both the coalescent and finite haplotype models ($k = 2, m = 10, n = 25$). **(left)** We compare m-clique cover to the original RDH criterion. The m-clique cover outperforms RDH under both models. This is large under the coalescent model, because it does not exclude multiple characters per tree edge. **(right)** Over a larger range of missing data rates, we compare *m-clique rooted* to other methods specific to $k = 2$. While *chordal* did not perform as well as *m-clique rooted*, it works for a non-nested set of instances and can be combined with *m-clique rooted* to obtain significantly better results.

For binary characters, under the coalescent model, we observed remarkable performance from *m-clique rooted*. In Fig. 4, it dramatically outperforms the other efficient methods. This was true for all missing data rates tested (Fig. 5). Since it is possible for the PI graph to be chordal and not have an m-clique, we also show in Fig. 4 improvement by combining *m-clique rooted* and *chordal*.

For data with more than two states, we frequently observed with *chordal* that the perfect phylogeny could be constructed efficiently. However, unlike the m-clique cover criterion, this is not with high probability for sufficiently large n, even under the *finite haplotype* model as observed in Fig. 5. The missing ingredient is ancestral taxa. If ancestral taxa were present in the population, then a chordal PI graph would have been a broader, faster, criterion to extend the RDH to k-state data. Unfortunately, this assumption is not generally valid.

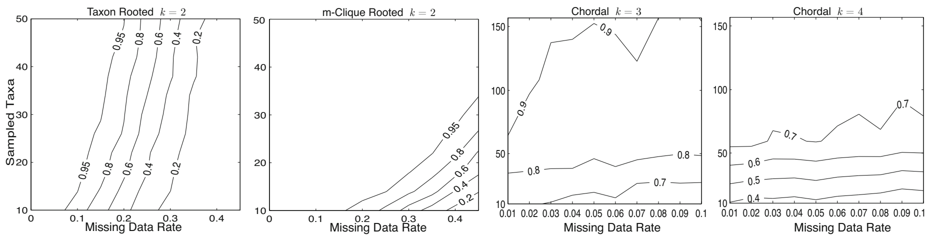


Fig. 5. (left pair) Contour plots comparing *taxon rooted* to our method *m-clique rooted*. Each plot consists of 2×10^5 draws from the coalescent model with $m = 10$. For fixed m, the probability of obtaining an m-clique increases much faster than the probability of obtaining a complete taxon. Intuitively, this is because when looking for a complete taxon, the taxa are examined independently. However, *m-clique rooted* combines multiple taxa to determine the root. **(right pair)** It is possible that the PI graph is chordal. For $k = 2$ this offers a small increase in the probability of success at very high missing data rates. For $k > 2$ the improvement is more pronounced. However, unlike an m-clique cover the data is not guaranteed to be chordal as $q \to 0$. For $k > 2$, the critical missing component are the ancestral taxa unobserved in the data.

Acknowledgements. We thank Dan Gusfield and three anonymous reviewers for helpful comments. TREE-POPPING is available at (http://treepopping.icsi.berkeley.edu/treepopping/). The other software necessary to reproduce the results presented here is also available from the authors. Partial support derived from CCF 1017580 and R01 HG002942. Both authors contributed equally to this extended abstract.

References

1. Agarwala, R., Fernandez-Baca, D.: A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. SIAM Journal of Computing 23(6), 1216–1224 (1994)
2. Blair, J.R.S., Peyton, B.: An introduction to chordal graphs and clique trees. In: George, A., Gilbert, J.R., Liu, J.W.H. (eds.) Graph Theory and Sparse Matrix Computation, pp. 1–29. Springer, Heidelberg (1993)
3. Buneman, P.: A characterisation of rigid circuit graphs. Discrete Mathematics 9(3), 205–212 (1974)
4. Ding, Z., Mailund, T., Song, Y.S.: Efficient whole-genome association mapping using local phylogenies for unphased genotype data. Bioinformatics 24(19), 2215–2221 (2008)
5. Dirac, G.A.: On rigid circuit graphs. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 25, 71–76 (1961)
6. Dress, A., Steel, M.: Convex tree realizations of partitions. Applied Math. Letters 5, 3–6 (1993)

7. Fulkerson, D.R., Gross, O.A.: Incidence matrices and interval graphs. Pac. J. of Math. 15(3), 835–855 (1965)
8. Golumbic, M.C.: Algorithmic graph theory and perfect graphs. North-Holland, Amsterdam (2004)
9. Gusfield, D.: The multi-state perfect phylogeny problem with missing and removable data: Solutions via integer-programming and chordal graph theory. In: Research in Computational Molecular Biology, pp. 236–252. Springer, Heidelberg (2009)
10. Gusfield, D., Frid, Y., Brown, D.: Integer programming formulations and computations solving phylogenetic and population genetic problems with missing or genotypic data. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 51–64. Springer, Heidelberg (2007)
11. Halperin, E., Karp, R.M.: Perfect phylogeny and haplotype assignment. In: RECOMB 2004: Proc.s of the 8th ann. Internat'l. Conf. on Comp. Mol. Bio., pp. 10–19. ACM Press, New York (2004)
12. Hudson, R.R.: Generating samples under a Wright-Fisher neutral model of genetic variation. Bioinformatics 18(2), 337–338 (2002)
13. Kannan, S., Warnow, T.: A fast algorithm for the computation and enumeration of perfect phylogenies when the number of character states is fixed. In: Proc. of the 6th Ann. ACM-SIAM Symp. on Disc. Alg., pp. 595–603. Society for Industrial and Applied Mathematics, Philadelphia (1995)
14. Lekkerkerker, C.G., Boland, J.C.: Representation of a finite graph by a set of intervals on the real line. Fundamenta Mathematicae 51, 45–64 (1962)
15. Lewis, J.G., Peyton, B.W., Pothen, A.: A fast algorithm for reordering sparse matrices for parallel factorization. SIAM J. on Sci. and Stat. Comp. 10(6), 1146–1173 (1989)
16. Li, N., Stephens, M.: Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. Genetics 165(4), 2213–2233 (2003)
17. McKee, T.A., McMorris, F.R.: Topics in intersection graph theory. SIAM Monographs on Discrete Mathematics (1999)
18. McMorris, F.R., Meacham, C.A.: Partition intersection graphs. Ars Combinatoria 16B, 135–138 (1983)
19. Meacham, C.A.: A manual method for character compatibility analysis. Taxon 30(3), 591–600 (1981)
20. Pe'er, I., Pupko, T., Shamir, R., Sharan, R.: Incomplete directed perfect phylogeny. SIAM J. Comput. 33(3), 590–607 (2004)
21. Pennington, G., Smith, C.A., Shackney, S., Schwartz, R.: Reconstructing tumor phylogenies from heterogeneous single-cell data. J. Bioinfo. and Comp. Bio. 5(2a), 407–427 (2007)
22. Satya, R., Mukherjee, A.: The undirected incomplete perfect phylogeny problem. IEEE/ACM Trans. on Comp. Bio. and Bioinfo. 5(4), 618–629 (2008)
23. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, Oxford (2003)
24. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. Journal of Classification, 91–116 (1992)
25. Stephens, M., Smith, N., Donnelly, P.: A new statistical method for haplotype reconstruction from population data. American Journal of Human Genetics 68, 978–989 (2001)
26. Sze, S.H., Lu, S., Chen, J.: Integrating sample-driven and pattern-driven approaches in motif finding. Algorithms in Bioinformatics, 438–449 (2004)

27. Tarjan, R., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM Journal on Computing 13, 566–579 (1984)
28. Warnow, T.J.: Tree compatibility and inferring evolutionary history. In: Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete algorithms, SODA 1993, pp. 382–391. Society for Industrial and Applied Mathematics, Philadelphia (1993)
29. Wu, Y.: Exact computation of coalescent likelihood under the infinite sites model. In: Măndoiu, I., Narasimhan, G., Zhang, Y. (eds.) ISBRA 2009. LNCS, vol. 5542, pp. 209–220. Springer, Heidelberg (2009)

Separating Metagenomic Short Reads into Genomes via Clustering
(Extended Abstract)

Olga Tanaseichuk[1], James Borneman[2], and Tao Jiang[1]

[1] Department of Computer Science and Engineering, UC Riverside, CA
[2] Department of Plant Pathology and Microbiology, UC Riverside, CA
{tanaseio,jiang}@cs.ucr.edu, borneman@ucr.edu

Abstract. The metagenomics approach allows the simultaneous sequencing of all genomes in an environmental sample. This results in high complexity datasets, where in addition to repeats and sequencing errors, the number of genomes and their abundance ratios are unknown. Recently developed next-generation sequencing (NGS) technologies significantly improve the sequencing efficiency and cost. On the other hand, they result in shorter reads, which makes the separation of reads from different species harder. In this work, we present a two-phase heuristic algorithm for separating short paired-end reads from different genomes in a metagenomic dataset. We use the observation that most of the l-mers belong to unique genomes when l is sufficiently large. The first phase of the algorithm results in clusters of l-mers each of which belongs to one genome. During the second phase, clusters are merged based on l-mer repeat information. These final clusters are used to assign reads. The algorithm could handle very short reads and sequencing errors. Our tests on a large number of simulated metagenomic datasets concerning species at various phylogenetic distances demonstrate that genomes can be separated if the number of common repeats is smaller than the number of genome-specific repeats. For such genomes, our method can separate NGS reads with a high precision and sensitivity.

1 Introduction

Metagenomics [1] is a new field of study that provides a deeper insight into the microbial world compared to the traditional single-genome sequencing technologies. Many well-known metagenomics projects use the whole genome shotgun sequencing approach in combination with Sanger sequencing technologies. This approach has produced datasets from the Sargasso Sea [2], Human Gut Microbiome [3] and Acid Mine Drainage Biofilm [4]. However, new sequencing technologies have evolved over the past few years. The sequencing process has been greatly parallelized, producing millions of reads with much faster speed and lower cost. Since NGS technologies are much cheaper, they allow sequencing to be performed at a much greater depth. The only drawback is that read length is reduced - NGS reads are usually of lengths 25-150 (Illumina/SOLiD) compared to 800-1000 bps in Sanger reads.

The primary goals of metagenomics are to describe the populations of microorganisms and to identify their roles in the environment. Ideally, we want to identify complete genomic sequences of all organisms present in a sample. However, metagenomic data is very complex, containing a large number of sequence reads from many species. The number of species and their abundance levels are unknown. The assembly of a single genome is already a difficult problem, complicated by repeats and sequencing errors which may lead to high fragmentation of contigs and misassembly. In a metagenomic data, in addition to repeats within individual genomes, genomes of closely related species may also share homologous sequences, which could lead to even more complex repeat patterns that are very difficult to resolve. A lot of research has been done for assembling single genomes [5,6,7,8]. But due to the lack of research on metagenomic assemblers, assemblers designed for individual genomes are routinely used in metagenomic projects [2,4]. It has been shown that these assemblers may lead not only to misassembly, but also severe fragmentation of contigs [9]. A plausible approach to improve the performance of such assemblers is to separate reads from different organisms present in a dataset before the assembly.

Many computational tools have been developed for separating reads from different species or groups of related species (we will refer to the problem as the clustering of reads). Some of the tools also estimate the abundance levels and genome sizes of species. These tools are usually classified as similarity-based (or phylogeny-based) and composition-based. The purpose of similarity-based methods is to analyze the taxonomic content of a sample by comparing of fragments against databases of known genes, proteins and genomic sequences [10,11,12]. The main drawback of similarity-based methods is that a large fraction of sequences may remain unclassified because of the absence of closely related sequences in the databases.

The second class of methods use compositional properties of the fragments (or reads). These methods are based on the fact that some composition properties, such as CG content and oligonucleotide frequencies are preserved across sufficiently long fragments of the same genome, and vary significantly between fragments from different organisms. K-mer frequency is the most widely used characteristics for binning. For example, the method in [13] utilizes the property that each genome has a stable distribution of k-mer frequencies for $k = 1..6$ in fragments as short as 1000 bps. The main challenge in the k-mer frequency approach is that these frequencies produce large feature vectors, which can be even larger than the sizes of fragments. Different methods have been proposed to deal with this problem. CompostBin [14], which uses hexamer frequencies, adopts a modified principle component analysis to extract the top three meaningful components and then cluster the reads based on principal component values. The work in [15] uses self-organizing maps to reduce dimensionality. In TETRA [16], z-scores are computed for tetranucleotide frequencies. MetaCluster 3.0 [17] uses Spearman Footrule distance between k-mer feature vectors. Another composition feature is used in TACOA [18]:

the ratio between observed oligonucleotide frequencies and expected frequencies given the CG content. The main limitation of composition based methods is that the length of fragments may significantly influence their performance. In general, these methods are not suitable for fragments shorter than 1000 bps [19].

AbundanceBin [20] is a recently developed tool for binning reads that uses an approach different from the above similarity and composition based techniques. It is designed to separate reads from genomes that have different abundance levels. It computes frequencies of all l-mers in a metagenomic dataset and, assuming that these frequencies come from a mixture of Poisson distributions, predicts the abundance levels of genomes and clusters l-mers according to their frequencies. Then reads are clustered based on the frequencies of their l-mers. This method is suitable for very short NGS reads. The limitation is that genomes whose abundance levels do not differ very much (within ratio 1:2) will not be separated.

In this paper, we present a two-phase heuristic algorithm for separating short paired-end reads from different organisms in a metagenomic dataset, called TOSS (*i.e.*, TOol for Separating Short reads). The basic algorithm is developed to separate genomes with similar abundance levels. It is based on several interesting observations about unique and repeated l-mers in a metagenomic dataset, which enables us to separate unique l-mers (each of which belongs to only one genome and is not repeated) from repeats (l-mers which are repeated in one or more genomes) at the beginning of the first phase of the algorithm. During the first phase, unique l-mers are clustered so that each cluster consists of l-mers from only one of the genomes. This is possible due to the observation that most l-mers are unique within a genome and, moreover, within a metagenomic dataset. During the second phase, we find connections between clusters through repeated regions and then merge clusters of l-mers that are likely to belong to the same organism. Finally, reads are assigned to clusters. In order to handle metagenomic datasets with genomes of arbitrary abundance ratios, we combine the method with AbundanceBin which attempts to separate l-mers from genomes with significantly different abundance levels. The integrated method works for very short reads, and is able to handle multiple genomes with arbitrary abundance levels and sequencing errors. We test the method on a large number of simulated metagenomic datasets for microbial species with various phylogenetic closeness according to the NCBI taxonomy [21,22] and show that genomes can be separated if the number of common repeats is less then the number of genome-specific repeats. For example, genomes of different species of the same genus often have a large number of common repeats and thus are very hard to separate. In the tests, our method is able to separate fewer than a half of groups of such closely related genomes. However, with the decrease in the fraction of common repeats, the ability to accurately separate genomes significantly increases. Due to the lack of appropriate short read clustering tools for comparison, we modify a well-known genome assembly software, Velvet [23], to make it behave like a genome separation tool and compare our clustering results with those of the modified Velvet.

The paper is organized as follows. In Section 2, we consider properties of l-mers in a metagenomic dataset and make several observations which form the intuition behind the algorithm, present the main algorithm, and extend the algorithm to handle arbitrary abundance ratios. Section 3 gives the performance evaluation on short reads and comparison with the well-known composition-based tool CompostBin on longer reads. Section 4 concludes the paper. Due to page limit, we omit the comparison with the modified Velvet on short reads, all pseudocode and some illustrative figures in this extended abstract. These omitted items can be found in the full paper [24].

2 Methods

2.1 Preliminaries

The algorithm we are going to present is based on l-mers from metagenomic reads. In this section, we will discuss some properties of l-mers that are important for our algorithm, and also make some important observations that lead to the intuition behind the algorithm.

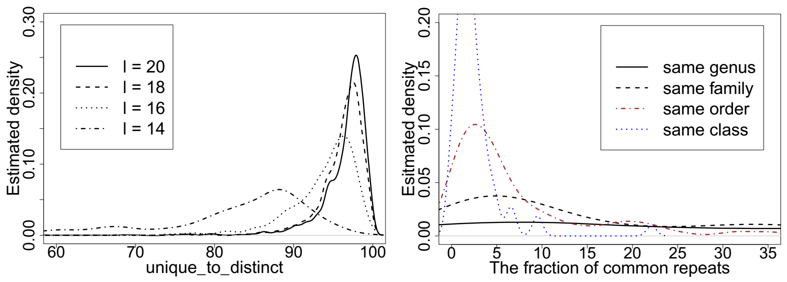
First, let us analyze the expected number of occurrence of l-mers in reads sequenced from a single genome of length G. Let the number of paired-end reads be N (which corresponds to $2N$ read sequences) and read length L. In shotgun sequencing projects, as well as NGS, the reads are randomly distributed across the genome. Since reads may begin at any positions of the genome with equal probability, Lander and Waterman suggested that the left ends of reads follow a Poisson distribution [25], which means that the probability for a read to begin at a given position of the genome is $\alpha = 2N/(G-L+1)$ and the number of reads starting at each position has a Poisson distribution with parameter α. Consider a substring w_i of length l that begins at the i-th position of the genome. Let $x(w_i)$ be the number of reads that cover this particular l-mer. Since there are $L - l + 1$ possible starting positions for such reads, $x(w_i)$ has a Poisson distribution with parameter $\lambda = \alpha(L - l + 1)$ (this parameter represents the effective coverage [25,26]). This analysis assume that the l-mer w_i occurs uniquely in the genome, but in general, an l-mer may occur multiple times. Suppose that an l-mer w has $n(w)$ copies in the genome located at positions $i_1, \ldots, i_{n(w)}$. Then the total number of reads containing w is $x(w) = \sum_{j=1}^{n(w)} x(w_{i_j})$. If we assume that a read covers at most one copy of w, then $x(w_{i_j}), j = 1, ..., n(w)$, are independent and identically distributed. So by the additivity property of the Poisson distribution, the total number of occurrences of w in the reads, $x(w)$, follows a Poisson distribution with parameter $\alpha(L-l+1)n(w)$. In [27], this model is used to find repeat families for a single genome, where a repeat family is a collection of l-mers that have the same number of copies in the genome.

In a metagenome, besides repeats that occur within individual genomes, genomes of different species may share common l-mers. Consider S genomes $g_j, j = 1, ..., S$, and assume that an l-mer w has $n_j(w)$ copies in each genome

$g_j, j = 1, ..., S$. Then the number of reads containing w is $x(w) = \sum_{j=1}^{S} \alpha_j(L - l + 1)n_j(w) = \sum_{j=1}^{S} \lambda_j n_j(w)$, where λ_j represents the effective coverage of genome g_j. Since sequencing depth is the same for all genomes, we will refer to it as the abundance. This model is quite difficult to use in practice because we do not know the number of genomes and their repeat structures, common repeats and abundance levels. A simplification of this model is used in AbundanceBin [20], by assuming that for large enough l, most l-mers appear only once in the genomes (not that in AbundanceBin, 20-mers are considered, compared to 12-mers considered in [27]). This allows the authors to estimate the abundance levels of genomes by modeling the abundance levels of the genomes as a mixture of Poisson distributions, where the parameters are the abundance levels of the genomes and their observed values are the counts of the l-mers (*i.e.*, the number of reads containing these l-mers). This approach works well if the abundance levels are sufficiently different. Also, it is applicable only if the above simplifying assumption holds. Below, we will discuss the validity of this assumption in real bacterial genomes and make three important observations about the distribution of l-mers. Before going into the details of the observations, let us introduce some notations. Consider two different genomes, g_1 and g_2, of lengths G_1 and G_2. Let n_1^{dist} denote the number of distinct l-mers in g_1, n_1^{uniq} the number of l-mers that have only one copy in g_1 (we will call them the unique l-mers in g_1) and n_1^{tot} the total number of l-mers in g_1 (including copies). Obviously $n_1^{tot} = G_1 - l + 1$. The notations for genome g_2 are defined similarly. Our first observation is the following: (1) Most of the l-mers in a bacterial genome are unique in this genome. To confirm it, we have computed the ratio of unique l-mers to distinct l-mers for all complete bacterial genomes downloaded from NCBI. Figure 1(a) shows the estimated density of this value. We can conclude that fraction of unique l-mers with $l = 20$ is between 96% and 100% for most of complete bacterial genomes.

In order to explain the second observation, let us introduce more notations. Let us consider l-mers from two genomes g_1 and g_2. Denote by n^{dist} the total number of distinct l-mers in both genomes together. We say that an l-mer is unique if it is present only in one genome and, moreover, unique in this genome. Then n^{uniq} denotes the number of unique l-mers in the genomes. Obviously, $n_1^{uniq} + n_2^{uniq} \geq n^{uniq}$, because some l-mers that are unique in one genome may not be unique in both genomes due to common repeats. Our second observation is concerned with the percentage of unique l-mers in a metagenome: (2) Most l-mers are unique in a metagenome if it consists of genomes of species separated by sufficiently large phylogenetic distances. The validation of this observation is discussed in the full paper [24].

From now on, by "unique l-mers" we will mean l-mers that appear only once in all the genomes. The remaining l-mers are repeats. We will further classify the repeats into two groups: *individual repeats* are l-mers which appear only in one genome (but have several copies) and *common repeats* are l-mers that appear



(a) The ratio of unique l-mers to distinct l-mers

(b) The ratio of common repeats to the total number of all distinct repeats

Fig. 1. (a) Estimated density functions of the fraction of unique l-mers in fully sequenced bacterial genomes for $l = 14, 16, 18, 20$. (b) Estimated density function of the ratio of the number of common repeats to the total number of all distinct repeats.

in at least two genomes. Our final observation is: (3) If genomes are separated by sufficient phylogenetic distances (they are at least from different families), then most of the repeats are individual repeats. In addition, the bigger is the phylogenetic distance between genomes, the fewer the common repeats. Figure 1(b) demonstrates the validity of this observation.

Our algorithm is based on these three observations. Since most of the l-mers are unique in a metagenome, we can cluster the unique l-mers by using their common membership in reads so that each cluster contains l-mers from only one genome in the first phase. The second phase of our algorithm uses the property that most of repeats are specific to an individual genome. This allows us to merge clusters using the repeated l-mers in the metagenome.

2.2 Finding Unique l-Mers

Before performing the first phase of the algorithm, which clusters the unique l-mers, l-mers have to be separated into unique l-mers and repeats. This is done by choosing a threshold value K for the counts of l-mers so that l-mers with counts less than K are most likely unique and the remaining are most likely repeats. Below, we discuss how to chose K.

First, consider error-free metagenomic reads of genomes with equal abundance levels. Let n be the number of distinct l-mers $w_1, w_2, ..., w_n$ with counts $x(w_1), x(w_2), ..., x(w_n)$. Let $n(i)$ be the number of distinct l-mers with counts i. As we discussed in the previous section, the unique l-mers follow a Poisson distribution and we may approximate the parameter of the Poisson distribution by the most frequent count of any l-mers because most l-mers are supposed to be unique. Then, given the estimated parameter, we can estimate the expected number of l-mers with counts i, $y(i)$. Figure 2 shows the count distributions of

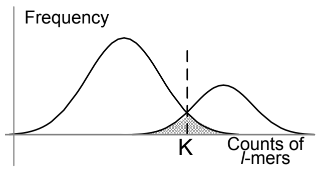


Fig. 2. Threshold choice for the separation of l-mers from different distributions

unique and non-unique l-mers, where the non-unique l-mers (*i.e.*, repeats) are assumed to be from a mixture Poisson distributions and the shaded area shows the expected rate of misclassified l-mers for the given threshold value K. In the figure, if we choose the threshold higher or lower, more repeats or unique l-mers would be undetected, respectively. Although we do not know the distribution of the repeats, we can see that the observed number of l-mers with count K is twice the expected number of unique l-mers with count K, and this ratio increases for count values greater than K. Based on this intuition, we can estimate the value of K. The details are given in Algorithm 1 of [24]. A similar approach is used to deal with sequencing errors, by finding a threshold value for counts of l-mers that separates unique l-mers and l-mers with errors.

The set U of unique l-mers is then used to construct a graph which can help detect more repeats and will be used to do the clustering. The nodes of the graph G correspond to the elements of U and there is an edge between two nodes if both l-mers are contained in a same read. To remove previously undetected repeats, we use the fact that nodes that correspond to truly unique l-mers cannot have more than $2(L - l)$ neighbors.

2.3 Clustering the Unique *l*-Mers

We use graph G described above to perform the clustering. The purpose is to obtain clusters so that each cluster contains unique l-mers from only one genome. Note that the number of such clusters for each genome can be large. We initialize the first cluster with the l-mers from a randomly selected read and then iteratively find sets of unclustered nodes that are connected to at least T nodes in the current cluster (the choice of T is discussed later in the subsection). It is important to note that the number of unique l-mers we can add at each step is limited by $2(L - (l + T) + 1)$, since we could add l-mers from both ends of a read. If we need to add more than this many l-mers at some step, it means that we have encountered true repeats that have not been removed and thus we stop expanding the current cluster. We also stop expanding the current cluster if no more nodes could be added. Then we go to the next iteration and construct the next cluster. For each such subsequent iteration, we initialize a new cluster with l-mers from some read that does not correspond to any of the current clusters. A read corresponds to a cluster if at least a half of its l-mers belong to the particular cluster. We create new clusters until there are no more unclustered reads left. At the end of clustering, we obtain a set of disjoint clusters of l-mers.

The paired-end information is then used to consolidate the clusters. The details are given in Algorithm 2 of the full paper [24].

Threshold T (the minimum required number of edges between an unclustered node and the nodes in a cluster so that the node can be added to this cluster) is chosen to make the expected number of coverage gaps less than one. Recall that the effective coverage is $Cov = 2N(L - (l + T) + 1)/(G - L + 1)$ and expected number of gaps is $2Ne^{-Cov}$ [26].

2.4 Merging Clusters and the Final Clustering of Metagenomic Reads

The goal of the second phase is to merge clusters obtained during the previous phase, based on the repeats and information provided by the paired-end reads. First, for each cluster C_i, we compute the set of repeats R_i that may potentially belong to the same genome as the unique l-mers in C_i. Each R_i consists of two types of l-mers. For each read corresponding to cluster C_i, it may contain some number of repeats. These repeated l-mers are assigned to the set R_i. For each read corresponding to C_i, we also consider its mate (in a paired-end read) and add to R_i all l-mers of the mate that have not been assigned to any clusters. Then for each pair of sets R_i and R_j, we find the intersection of these sets, R_{ij}. Then, we build a weighted graph F, where nodes correspond to clusters C_i and the weight of an edge (i, j) equals the size of set R_{ij}. Finally, the clusters are merged by using the algorithm MCL [28] on the graph F. MCL is an efficient algorithm for clustering sparse weighted graphs and ideal for our situation. To avoid confusion, we will call clusters produced by MCL the *m-clusters*. MCL has a parameter (we denote it by r), corresponding to granularity of clusters. We use an iterative algorithm to find the best parameter so that the m-clusters are big enough (in terms of the number of l-mers contained in each m-cluster) and the total weight of connections between elements within an m-cluster is higher than the total weight of connections between two different m-clusters. Let us call m-cluster that satisfy the first property *big*, and a subset of big m-clusters that satisfy the second property (with respect to all other big m-clusters) *valid*. We start with a parameter r which corresponds to a high granularity and evaluate the resultant clusters in terms of size and validity. Based on the evaluation, we either decrease the parameter to have less granularity or choose the current value of r as the parameter for MCL. We obtain final clusters of the unique l-mers by merging clusters that belong to the same m-cluster (see Algorithm 3 in the full paper [24] for details).

Now we discuss how to define big and valid m-clusters. The minimum size of a big m-cluster is specified by the user based on the minimum expected length of a genome. Valid m-clusters are chosen from big m-clusters in the following way. Let W_{jj} and W_{ii} be the total weights of the connections within each of the m-clusters j and i, and W_{ij} the total weight of the connections between these two m-clusters. The big m-cluster i is defined to be valid if for every other big m-clusters j, the inequality $\sqrt{\frac{W_{ij}}{WiiWjj}} > 10^{-3}$ holds. The threshold of 10^{-3} is chosen empirically.

In the final step of the algorithm, the reads are assigned to the resultant clusters of unique l-mers. Iterative algorithm is used to assign the reads. At the first step, each reads that correspond to some cluster is assigned to this cluster. During the second step, unassigned reads that have assigned mates are assigned to the same clusters as their mates. In the third step, for each cluster of unique l-mers we add all the l-mers from the reads assigned to the cluster. We iteratively repeat the three steps for the unassigned reads until no more reads can be assigned. If the read correspond to several clusters, we assign it to one of the clusters.

2.5 Handling Genomes with Arbitrary Abundance Levels

We would like to extend the above algorithm to metagenomic data containing genomes with different abundance levels. If the abundance level difference is not significant, the above algorithm would still work well. In this case, the number of wrongly determined unique l-mers and repeats in the first phase of the algorithm may slightly increase, but the clustering of l-mers based on their counts using the Poisson mixture model may incur a significantly higher drop of performance. For genomes with significantly different abundance levels, it makes sense to first separate reads according to genome abundance levels. Otherwise, repeats from genomes with lower abundance levels will not be detected, which could lead to a significant increase of granularity in the clustering result produced by the first phase of the above algorithm. For this reason, we propose to use the algorithm AbundanceBin [20] for the initial abundance-based binning of reads. Then we run the first phase of our method for each of the subsets of reads. For the second phase, we use all the reads to find the connections between clusters so that connections between clusters from genomes with low abundance levels are properly recovered, but MCL is performed on each subset separately.

A key question is what ratios of abundance levels should be considered as significant? This ratio depends on the actual values of abundance levels and also on the sizes of the genomes. Given abundance levels λ_1 and λ_2 ($\lambda_1 < \lambda_2$), genome sizes G_1 and G_2, and a threshold K for classifying l-mers into the two genomes based on count frequencies, we can estimate the expected rate of misclassified l-mers from the count distributions of the l-mers in these two genomes as discussed in Section 2.2. More specifically, the shaded area in Figure 2 represents the expected fraction of misclassification for two distributions. The number of l-mer in this area is $l_2 \sum_{i=1}^{K-1} \frac{\lambda_2^i e^{-\lambda_2}}{i!} + l_1 \sum_{i=K}^{Max} \frac{\lambda_1^i e^{-\lambda_1}}{i!}$. So, we first use AbundanceBin to predict the parameters of count distributions (*i.e.*, the abundance ratios and genome sizes) and then compute the expected rate of misclassification. If this rate is unacceptable (we used 3% as the threshold in the experiments), it means that the abundance levels are not significantly different and thus we do not run AbundanceBin.

3 Experimental Results

We test the performance of our algorithm on a variety of synthetic datasets with different numbers of species, phylogenetic distances between species, abundance ratios and sequencing error rates. Although simulated datasets do not capture all characteristics of real metagenomic data, there are no real benchmark datasets for NGS metagenomic projects and thus they are the only available option. Also, to the best of our knowledge, there are no algorithms that are designed specifically for separating short NGS reads from different genomes. We compare the performance of our algorithm with the well-known composition-based method CompostBin [14] on simulated metagenomic Sanger reads. We also apply the algorithm to a real metagenomic dataset obtained from gut bacteriocytes of the glassy-winged sharpshooter and achieve results consistent with the original study [29].

3.1 Simulated Data Sets

We use MetaSim [30] to simulate paired-end Illumina reads for various bacterial genomes to form metagenomic datasets. MetaSim is a software for generating metagenomic datasets with controllable parameters, such as the abundance level of each genome, read length, sequencing error rate and distribution of errors. Thus, it can be used to simulate different sequencing technologies and generate reads from available completely sequenced genomes (for example, those in the NCBI database). In our experiments, paired-end reads of length 80 bps are considered, with the mean insert size 500 bps and deviation 20 bps. The number of reads for each experiment is adjusted to produce sufficient coverage depth (ranging between 15 and 30). The sequencing error model is set according to the error profile of 80 bps Illumina reads.

The first experiment is designed to test the performance of our method on a large number of datasets of varying phylogenetic distances. For this experiment, we create 182 synthetic datasets of 4 categories. Each dataset of the first category contains genomes from the same genus but different species. Datasets in the second category consist of genomes from the same family but different genera, datasets in the third category involve genomes from the same order but different families, and datasets in the fourth category involve genomes from the same class but different orders. Genomes in each test are randomly chosen according to a category of phylogenetic distances and assumed to have the same abundance levels. The number of genomes in the datasets varies from 2 to 10 and depends on the number of available complete sequences for each taxonomic group and on the level of the group. Tests on genomes from the same genus typically involve 2 to 4 genomes since such genomes are similar to each other and hard to separate, while tests on genomes from the same class may involve up to 10 genomes. Totally, we have 79 experiments concerning a genus, 66 concerning a family, 29 concerning an order, and 8 concerning a class. These datasets involve 515 complete genomes from the NCBI.

We also performed some small-scale experiments to test the performance on genomes with different abundance levels and on reads with sequencing errors. For each of the experiments, we choose 10 random sets of genomes from the 182 datasets. For each set of genomes, two metagenomic dataset are simulated, one with abundance ratio 1:2 and and the second with the error model but abundance ratio 1:1. Finally, we test the performance of the combination of our algorithm and AbundanceBin on a dataset of 4 genomes with abundances 1:1:4:4.

3.2 Performance Evaluation

To evaluate the results of clustering, there are a number of factors that should be considered. First of all, we would like most of the reads from each genome to be located in one cluster. In other words, each genome should correspond to a unique cluster that contains most of its reads. We say that a genome has been *broken* if there is no cluster that contains more than a half of all its reads. It may happen that several genomes correspond to the same cluster. In this case, we assign the cluster to all the genomes, and say that the genomes are not separated. We will measure the performance of our algorithm in terms of pairwise separability. During the separability analysis, we remove broken genomes from consideration. Besides separability, we are interested in the precision and sensitivity of our algorithm on the separated genomes. Since we assign a genome to the cluster that has most of its reads, it is also interesting to know how many of its reads are wrongly assigned to other clusters. We call this *sensitivity.* One way to estimate sensitivity is to compute how many reads are correctly assigned to each cluster and divide it by the total number of reads that should be in this cluster. Here, true positives are the reads from all genomes located in this cluster. However, consider the case when we have two genomes in a cluster, of lengths 1 Mbps and 5 Mbps respectively. Then, even if sensitivity is very low for the first genome, the overall sensitivity (for all genomes in the cluster) will not be significantly affected. Another way to normalize sensitivity is by computing sensitivity for each genome in the cluster separately and then to find the average of these sensitivities. We use the second approach. To compute *precision* of a cluster, we find the ratio of the reads that are wrongly assigned to the cluster to the total number of reads in the cluster.

To summarize the results for a set of experiments, we compute separability based on the total number of pairs of genomes in all the experiments. For the precision and sensitivity, we take the average values for all the clusters from all the experiments.

3.3 Experiments on Genomes Separated by Different Phylogenetic Distances

Our experimental results on metagenomic datasets containing genomes with different phylogenetic distances are summarized in Table 1. For genomes from the same genus, separability rate is 45%. It increases to 77% for genomes from the same family and more than 97% for higher level taxonomic categories. For separated pairs of genomes, our sensitivity increases from 90% for genomes from

Table 1. Performance on pairs of genomes with different phylogenetic distances

	# of genomes	# of pairs	Broken	Separated	Sensitivity		Precision	
					mean	stdv	mean	stdv
Species	229	184	3	83	90.38	8.71	95.55	5.96
Genus	171	147	2	113	93.40	9.39	97.07	5.52
Family	80	79	2	75	94.98	6.56	97.46	5.86
Order	35	71	0	70	95.14	4.87	97.79	2.49

Table 2. Performance of the method on synthetic datasets with and without sequencing errors. The third row assumes the abundance ratio 1:2.

	# of genomes	# of pairs	Broken	Separated	Sensitivity	Precision
IdentAbund, error-free	24	18	0	18	93.48	96.08
IdentAbund, with errors	24	15	2	15	96.84	98.03
DiffAbund, error-free	24	18	0	17	91.00	98.25

the same genus to 95% for genomes from the same order. The range of precision is from 95% to 98%. These results are consistent with our expectation for correlation between separability and phylogenetic distance.

3.4 Handling Sequencing Errors

Our approach for handling sequencing errors is very simple: we filter out l-mers with counts lower than a certain threshold, since these infrequent l-mers are likely to contain errors. However, there is a simple intuition behind it. We can aggressively remove potential errors without attempting to correct them or being afraid to lose important information, assuming that the genomes are sufficiently covered by the reads. Note that we could be more aggressive than genome assemblers in throwing out infrequent l-mers here because (i) when the genomes are sufficiently covered, the filtration will not lead to many more gaps, and (ii) we are less concerned with the fragmentation of genomes.

In Table 2, we summarize our experimental results on pairs of genomes with and without sequencing errors. We can see that our method is able to separate more pairs of genomes when the reads are error-free. However, when broken genomes are discounted, the method actually achieves a slightly higher sensitivity and precision on data with errors.

3.5 The Issue of Abundance Levels

In this section, we analyze the ability of our method to separate genomes with different abundance levels. First, we test our algorithm (without any modification) on pairs of genomes with abundance ratio 1:2 and compare the results with those on the same set of pairs of genomes but with identical abundance levels.

Table 3. Performance on synthetic datasets with abundance ratio (1:1:4:4)

	# of genomes	# of pairs	Broken	Separated	Sensitivity	Precision
DiffAbund	4	6	0	6	97.42	99.81
IdentAbund	4	6	0	6	92.10	93.81

The results are summarized in the last row of Table 2. We can see that sensitivity slightly drops on genomes with different abundance levels, but precision actually improves a little.

We also test the performance of a combination of AbundanceBin and our algorithm on a set of four genomes with abundance levels $(1:1:4:4)$ and compare its result with that of our (original) algorithm on the same set of genomes with identical abundance levels. The results are summarized in Table 3. As we can see, the result on data with varying abundance levels is actually better. Sensitivity and precision increase from 92% and 93% on data with identical abundance levels to 97% and 99% on data with varying abundance levels. In order to explain this (somewhat counter-intuitive) phenomenon, we analyzed intermediate results, and found that two of the six pairs of genomes, (1,3) and (2,4), have high percentages of common repeats. These common repeats negatively affected the result on data with identical abundance levels. However, they did not cause any trouble for the test on data with varying abundance levels since for both pairs, reads from different genomes were separated by AbundanceBin early on due to the difference in their abundance levels.

3.6 Comparison with CompostBin

In this section, we compare the performance of our algorithm with a composition-based binning algorithm, CompostBin [14]. Note that composition-based methods require sufficiently long reads while TOSS is designed to separate short NGS reads. On the other hand, our method requires a high coverage depth. To compare the performance with CompostBin, we use the simulated paired-end Sanger reads of length 1000 bps provided in [14]. We slightly adapt our method to handle longer reads and lower coverage. In particular, we use a higher threshold in the prediction of unique l-mers. Also, we cut the Sanger reads into fragments of length 80 bps before constructing the graph of unique l-mers in order to minimize memory usage. The linkage information of the fragments belonging to a same read will be recovered and taken advantage of later in the cluster merging phase. Normalized error rates (as defined in [14]) for our algorithm and for CompostBin are reported in Table 4. Note that in the last three datasets, the average coverage of genomes with lower abundance levels (not shown in the table) is close to 1 and, therefore, is insufficient for our algorithm. In addition, we simulate Illumina reads from the same sets of genomes with a coverage depth between 15 and 30. Normalized error rates for these datasets are shown in the last column of Table 4. The highest error rates of our algorithm on Sanger and Illumina reads are 4.74% and 4.92% respectively, and are less than 10% for CompostBin. For some of the

Table 4. Comparison with CompostBin on the datasets described in [14]. Note that some datasets involve genomes that are separated at several different taxonomic levels.

Species	Ratio	Phylogenetic Distance	CompostBin Error, %	TOSS (Sanger) Error, %	TOSS (Illumina) Error, %
Bacillus halodurans & Bacillus subtilis	1:1	Species	6.48	1.05	1.38
Gluconobacter oxydans& Granulibacter bethesdensis	1:1	Genus	3.39	4.72	4.92
Escherichia coli & Yersinia pestis	1:1	Genus	10.00	3.11	2.58
Methanocaldococcus jannaschii & Methanococcus maripaludis	1:1	Family	0.51	0.22	0.01
Pyrobaculum aerophilum & Thermofilum pendens	1:1	Family	0.28	1.05	0.01
Gluconobacter oxydans & Rhodospirillum rubrum	1:1	Order	0.98	4.74	0.01
Gluconobacter oxydans, Granulibacter bethesdensis & Nitrobacter hamburgensis	1:1:8	Family and Order	7.7	-	6.45
Escherichia coli, Pseudomonas putida & Bacillus anthracis	1:1:8	Order and Phylum	1.96	-	0.15
Escherichia coli, Pseudomonas putida, Thermofilum pendens, Pyrobaculum aerophilum, Bacillus anthracis & Bacillus subtilis	1:1: 1:1: 2:14	Species, Order, Family, Phylum, and Kingdom	4.52	-	0.80

Sanger datasets, the performance of our algorithm is slightly worse compared to CompostBin and for the others, it is slightly better. The performance of our algorithm on the corresponding Illumina datasets is better in most of the cases. Clearly, the higher coverage depths in Illumina datasets helped. A high coverage depth is essential for the accurate prediction of unique and repeated l-mers in the preprocessing phase of our algorithm.

3.7 Performance on a Real Dataset

A metagenomic dataset obtained from gut bacteriocytes of the glassy-winged sharpshooter, *Homalodisca coagulata*, is known to consist of (Sanger) reads from *Baumannia cicadellinicola*, *Sulcia muelleri* and some miscellaneous unclassified reads [29] and studied in [14]. We apply our algorithm, adapted to handle Sanger reads as discussed in Section 3.6, to the dataset. As in [14], we only measure our ability to separate the reads from *Baumannia cicadellinicola* and *Sulcia muelleri*. The sensitivity of the classification achieved by our algorithm is 92.21% and the normalized error rate is 1.59%, which is lower than the normalized error rate of 9.04% achieved by CompostBin as reported in [14].

4 Implementation and Future Work

Our algorithm called TOSS was coded in C. Its running time and memory requirement depend on the total length of all the genomes present in a metagenomic dataset and on the number of reads. The first phase of the algorithm is the most time and memory consuming. In this phase, a graph of l-mers is constructed and the clustering of unique l-mers is performed. The size of the graph is proportional to the total size of the genomes and 0.5 GB of RAM is required for every million bases of the genomes. In the experiments, we ran the algorithm on a single CPU with 2.8GHz AMD machine and 64GB RAM. Each of the small-scale tests involving 2-4 genomes of total length of 2-6 Mbps was completed within 1-3 hours and required 2-4 GB of RAM. A test on 15 genomes with the total length of 40 Mbps ran for 14 hours and required 20GB of RAM.

In future work, we plan to explore the compositional properties of the clusters of unique l-mers and try to improve the performance of our algorithm by combining the compositional properties with the distribution of l-mers in reads.

Acknowledgments. The research is supported in part by NSF grant IIS-0711129 and NIH grant AI078885.

References

1. Handelsman, J., Rondon, M.R., Brady, S.F.: Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. Chemistry & biology 5 (October 1998)
2. Venter, J.C., Remington, K., Heidelberg, J.F., et al.: Environmental Genome Shotgun Sequencing of the Sargasso Sea. Science 304, 66–74 (2004)
3. Gill, S.R., Pop, M., DeBoy, R.T., et al.: Metagenomic Analysis of the Human Distal Gut Microbiome. Science 312, 1355–1359 (2006)
4. Tyson, G.W., Chapman, J., Hugenholtz, P., et al.: Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature 428, 37–43 (2004)
5. Chaisson, M.J., Pevzner, P.A.: Short read fragment assembly of bacterial genomes. Genome research 18, 324–330 (2008)
6. Warren, R.L., Sutton, G.G., Jones, S.J.M., et al.: Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23, 500–501 (2007)
7. Dohm, J.C., Lottaz, C., Borodina, T., Himmelbauer, H.: SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Research 17, 1697–1706 (2007)
8. Simpson, J.T., Wong, K., Jackman, S.D., et al.: ABySS: A parallel assembler for short read sequence data. Genome Research 19, 1117–1123 (2009)
9. Charuvaka, A., Rangwala, H.: Evaluation of Short Read Metagenomic Assembly. Tech. Rep. GMU-CS-TR-2010-9 (2010)
10. Chakravorty, S., Helb, D., Burday, M., et al.: A detailed analysis of 16s ribosomal RNA gene segments for the diagnosis of pathogenic bacteria. J. Microbiol Methods 69(2) (2007)
11. Huson, D.H., Auch, A.F., Qi, J., et al.: MEGAN analysis of metagenomic data. Genome research 17, 377–386 (2007)
12. Krause, L., Diaz, N.N., Goesmann, A., et al.: Phylogenetic classification of short environmental DNA fragments. Nucleic Acids Research 36, 2230–2239 (2008)
13. Zhou, F., Olman, V., Xu, Y.: Barcodes for genomes and applications. BMC Bioinformatics 9(1), 546+ (2008)

14. Chatterji, S., Yamazaki, I., Bai, Z., et al.: Compostbin: a dna composition-based algorithm for binning environmental shotgun reads. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 17–28. Springer, Heidelberg (2008)
15. Chan, C.-K., Hsu, A., Halgamuge, S., Tang, S.-L.: 'Binning sequences using very sparse labels within a metagenome. BMC Bioinformatics 9(1) (2008)
16. Teeling, H., Waldmann, J., Lombardot, T., et al.: TETRA: a web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in DNA sequences. BMC Bioinformatics 5, 163+ (2004)
17. Leung, H.C.M., Yiu, S.M., Yang, B., et al.: A robust and accurate binning algorithm for metagenomic sequences with arbitrary species abundance ratio. Bioinformatics 27, 1489–1495 (2011)
18. Diaz, N., Krause, L., Goesmann, A., et al.: TACOA - Taxonomic classification of environmental genomic fragments using a kernelized nearest neighbor approach. BMC Bioinformatics 10(1), 56+ (2009)
19. Bentley, S.D., Parkhill, J.: Comparative genomic structure of prokaryotes. Annual Review of Genetics 38, 771–791 (2004)
20. Wu, Y.-W., Ye, Y.: A novel abundance-based algorithm for binning metagenomic sequences using l-tuples. In: Berger, B. (ed.) RECOMB 2010. LNCS, vol. 6044, pp. 535–549. Springer, Heidelberg (2010)
21. Wheeler, D.L., Barrett, T., Benson, D.A., et al.: Database resources of the National Center for Biotechnology Information. Nucleic Acids Research 35 (January 2007)
22. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., et al.: GenBank. Nucleic acids research 37, D26–D31 (2009)
23. Zerbino, D.R., Birney, E.: Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Research 18, 821–829 (2008)
24. Tanaseichuk, O., Borneman, J., Jiang, T.: Separating metagenomic short reads into genomes via clustering (2011) (manuscript), `http://www.cs.ucr.edu/~tanaseio/metagenomic-full.pdf`
25. Lander, E.S., Waterman, M.S.: Genomic mapping by fingerprinting random clones: a mathematical analysis. Genomics 2, 231–239 (1988)
26. Wendl, M., Waterston, R.: Generalized gap model for bacterial artificial chromosome clone fingerprint mapping and shotgun sequencing. Genome Res. 12(1), 1943–1949 (2002)
27. Li, X., Waterman, M.S.: Estimating the Repeat Structure and Length of DNA Sequences Using l-Tuples. Genome Research 13, 1916–1922 (2003)
28. van Dongen, S.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (May 2000)
29. Wu, D., Daugherty, S.C., Van Aken, S.E., et al.: Metabolic Complementarity and Genomics of the Dual Bacterial Symbiosis of Sharpshooters. PLoS Biol 4, e188+ (2006)
30. Richter, D.C., Ott, F., Auch, A.F., et al.: MetaSim: a Sequencing Simulator for Genomics and Metagenomics. PLoS ONE 3, e3373+ (2008)

Finding Driver Pathways in Cancer: Models and Algorithms

Fabio Vandin[1,2], Eli Upfal[1,2], and Benjamin J. Raphael[1,2]

[1] Department of Computer Science, and
[2] Center for Computational Molecular Biology, Brown University, Providence, RI
{vandinfa,eli,braphael}@cs.brown.edu

Abstract. Cancer sequencing projects are now measuring somatic mutations in large numbers of cancer genomes. A key challenge in interpreting these data is to distinguish *driver mutations*, mutations important for cancer development, from *passenger* mutations that have accumulated in somatic cells but without functional consequences. A common approach to identify genes harboring driver mutations is a *single gene test* that identifies individual genes that are mutated in a significant number of cancer genomes. However, the power of this test is reduced by the mutational heterogeneity in most cancer genomes and by the necessity of estimating the *background mutation rate* (BMR). We investigate the problem of discovering *driver pathways*, groups of genes containing driver mutations, directly from cancer mutation data and without prior knowledge of pathways or other interactions between genes. We introduce two generative models of somatic mutations in cancer and study the algorithmic complexity of discovering driver pathways in both models. We show that a single gene test for driver genes is highly sensitive to the estimate of the BMR. In contrast, we show that an algorithmic approach that maximizes a straightforward measure of the mutational properties of a driver pathway successfully discovers these groups of genes without an estimate of the BMR. Moreover, this approach is also successful in the case when the observed frequencies of passenger and driver mutations are indistinguishable, a situation where single gene tests fail.

1 Introduction

Cancer is a disease driven in part by somatic mutations that accumulate during the lifetime of an individual. These mutations include single nucleotide substitutions, small indels, and larger copy number aberrations and structural aberrations. A key challenge in cancer genomics is to distinguish *driver mutations*, mutations important for cancer development, from random *passenger mutations* that have accumulated in somatic cells but do not have functional consequences. Recent advances in DNA sequencing technologies allow the measurement of somatic mutations in large numbers of cancer genomes. Thus, a common approach to identify driver mutations, and the driver genes in which they reside, is to identify genes with recurrent mutations in a large cohort of cancer patients. The standard approach to identify such recurrently mutated genes is to perform a *single gene test*, in which individual genes are tested to determine if their observed frequency of mutation is significantly higher than expected [11,5,12]. This approach has identified a number of important cancer genes, but has not revealed all of the driver mutations and driver genes in individual cancers.

There are two difficulties with the identification of driver genes by a single gene test of recurrent mutation. First, the test requires a reasonable estimate of the *background mutation rate* (BMR), which quantifies the accumulation of passenger mutations. Obtaining such an estimate is not a straightforward task, as the BMR is not just the rate of somatic mutation per nucleotide per cell generation, but also must account for selection and clonal amplification in the somatic evolution of a tumor [7,11]. Second, it is widely observed that there is extensive mutational heterogeneity in cancer, with mutations occurring in different genes in different patients. This mutational heterogeneity is a consequence of both the presence of passenger mutations in each cancer genome, and the fact that driver mutations typically target genes in cellular signaling and regulatory pathways [8,16]. Since each of these pathways contains multiple genes, there are numerous combinations of driver mutations that can perturb a pathway important for cancer. This mutational heterogeneity inflates the number of patients required to distinguish passenger from driver mutations, as rare driver mutations may not be observed at frequencies above the background. Thus, a common alternative to single gene tests is to test the recurrence of mutations in groups of genes derived from known pathways [6,2] or genome-scale gene interaction networks [3,13]. However, these approaches require prior knowledge of the interactions between genes/proteins, and this knowledge is presently far from complete. Moreover, pathway/network based approaches typically also require an estimate of the BMR.

The availability of somatic mutation data from increasing numbers of cancer patients motivates the question of whether it is possible to identify *driver pathways*, groups of genes with recurrent driver mutations, *de novo*; i.e. without prior knowledge of interactions between genes/proteins. At first glance, this seems implausible because there are an enormous number of possible sets of genes to consider. For example, there are more than 10^{25} sets of 7 human genes. However, we previously showed that mild additional constraints on the expected patterns of somatic mutations considerably reduce the number of gene sets to examine, and make *de novo* discovery of driver pathways possible [14]. These constraints are consistent with the current understanding of the somatic mutational process of cancer [9,16]. In particular, we assume that an important cancer pathway should be perturbed in a large number of patients. Thus, given genome-wide measurements of somatic mutations, we expect that a driver pathway will have high *coverage*: i.e. most patients will have a mutation in some gene in the pathway. Second, a driver mutation in a single gene of the pathway is often assumed to be sufficient to perturb the pathway. Combined with the fact that driver mutations are relatively rare, most patients exhibit only a single driver mutation in a pathway. Thus, we expect that the genes in a pathway exhibit a pattern of *mutually exclusive* driver mutations, where driver mutations are observed in exactly one gene in the pathway in each patient [17].

Note that the exclusivity constraint is assumed only for driver mutations in the *same* pathway. As a cancer genome likely has multiple driver pathways, the exclusivity assumption does not preclude the presence of co-occurring, and possibly cooperative, mutations, examples of which are known [15,4]. It is also possible that co-occurring mutations are necessary to perturb a pathway. In this case, there will likely remain a large subset of genes in the pathway whose mutations are exclusive, e.g. a subset obtained by removing one gene from each co-occurring pair. The identification of these subsets of genes by the approaches described here can be a starting point to later identify the other genes with co-occurring mutations.

1.1 Our Contribution

This work proposes a mathematical framework to study the problem of *de novo* discovery of driver genes and pathways. We define two generative models of driver mutations in cancer and study the algorithmic complexity of the discovery problem in each of the models, both analytically and in simulations. The two generative models differ in how conditioning on a sample being from a cancer patient affects the ratio between the driver and passenger mutation probabilities in that sample. While the difference is relatively small, it has a major implication on the practicality of the standard single gene test for identifying the driver genes. In the first model we prove a bound on the number of patients required to detect all driver genes with high probability using a single gene test, while in the second model it is not possible to identify the driver genes using such a test for *any* number of patients.

Next, we study a weight function on sets of genes that quantifies the coverage and exclusivity properties of a driver pathway. We introduced this function in [14], and showed that finding sets with high weight provides an alternative approach for identifying driver mutations. Here, we prove that for both generative models, when mutation data from enough patients is available, the weight function is monotone in the number of discovered driver genes and is maximized by the driver pathway. Based on this observation we prove that a simple greedy algorithm identifies the driver pathways with high probability. This improves the result in [14], where we showed that the discovery problem is NP-hard for arbitrary mutation data and that a greedy algorithm performs well under different conditions that did not arise from a generative model of the data. We also show that our earlier Markov Chain Monte Carlo (MCMC) approach for identifying the driver pathways rapidly converges to the driver pathway in both generative models, thus improving the convergence result of [14] for arbitrary mutation data. These results show that we can identify driver pathways *without* an estimate of the background mutation rate (BMR), giving a more reliable and robust solution for the problem.

We complement our analytical results with experiments on simulated data from the first model. We compare the number of patients required to identify driver genes using the single gene test with the number required using the greedy algorithm that maximizes the weight function. We show that the number of patients is similar when a perfect estimate of the BMR is available, but that the greedy algorithm requires a smaller number of patients when the estimate of the BMR deviates from its real value. Our analytical and experimental results help characterize the limitations of detecting driver genes and pathways under reasonable models of somatic mutation.

2 Stochastic Models for Somatic Mutations in Cancer

In this section we introduce two stochastic models for somatic mutations in cancer. In both models driver mutations occur in *sets* of genes, which we refer to as *driver pathways*. Passenger mutations occur randomly across all genes. We assume that mutations have been measured in n genes in a collection of m cancer patients, and represent the somatic mutations as a $m \times n$ binary mutation matrix A. The entry A_{ig} in row i and column g is equal to 1 if gene g is mutated in patient i, and it is 0 otherwise. Let $\mathcal{G}$ be the set of all columns (genes). In both models, we assume that the mutation matrix contains a *driver pathway*: a subset $\mathcal{D} \subseteq \mathcal{G}$ of genes, with $|\mathcal{D}| = k$, such that in each patient *exactly one* of the genes of $\mathcal{D}$ contains a driver mutation. Thus, a driver pathway $\mathcal{D}$ exhibits the properties of high *coverage* – every patient has a mutation in a gene in $\mathcal{D}$ – and *mutual exclusivity* – no patient has a driver mutation in more than one gene in $\mathcal{D}$. In both models, random *passenger* mutations occur at random in all genes, including genes in $\mathcal{D}$. The difference between the two models is in the relative mutation rates in driver and passenger genes. In the following we consider the case in which the mutation matrix contains only one driver pathway. However, our results can be generalized to the case of multiple disjoint driver pathways. In particular the following iterative procedure identifies all driver pathways using our algorithms: once we identify a driver pathway, we remove its genes from the mutation matrix, and look for driver pathways in the reduced mutation matrix.

Following the hypothesis that cancer is triggered by a mutation in a driver gene, the sample of cancer patients can be viewed as a subset of a larger initial population. The genome of each member of the initial population was subject to random mutations, where each gene was mutated independently, and our sample is the subset of the initial population with a driver mutation in a gene of $\mathcal{D}$.

The first stochastic model captures the above intuition by modeling the distribution of mutations in patients as independent with fixed probability q, conditioning on having a driver mutation. The mutation matrix A is generated by the following process: in each row (patient) we choose one gene $d \in \mathcal{D}$ uniformly at random to contain the driver mutation, and set the corresponding entry A_{id} to 1. All other entries at that row are set to 1 with probability $q < 1$ and to 0 otherwise, and all events are independent. We call the parameter q the *passenger mutation probability*[1], as it is the probability that a gene contains a passenger mutation. We denote the model above as the `D>P` model.

[1] Note that q is greater than the BMR, since it is the probability that a *gene* has a passenger mutation. For example, estimates of the BMR are typically $\approx 10^{-5}$, and since the length of most genes is around 10^4, we have that $q \approx 10^{-1}$.

A possible limitation of the D>P model is that it implies a conditional distribution in which driver genes have higher expected frequency of mutation than the passenger genes (thus the name D>P model) in a cohort of patients. In practice the driver pathway could contain dozens of genes, and some of them may have rare driver mutations. Thus the expected frequency of mutation of some genes in $\mathcal{D}$ may be indistinguishable from the expected frequency of mutation of some passenger genes. To examine this situation we introduce a second model, which we call the D=P model, in which all genes in $\mathcal{G}$ are mutated with the same probability in the patients, regardless of whether they are driver or passenger genes. Of course, this is a "worst case" model, as any cancer cohort with a reasonable number of patients will have some driver genes mutated at appreciable frequency. Nevertheless, we study the D=P model to consider the limits of driver pathway identification. The mutation matrix A in the D=P model is generated by the following process: in row (patient) i an entry A_{id} is chosen uniformly at random for $d \in \mathcal{D}$ and is set to 1. All other entries $A_{id'}$ for $d' \in \mathcal{D}$ are set to 1 with probability $r = \frac{qk-1}{k-1}$, and all entries A_{ig}, for $g \in \mathcal{G} \setminus \mathcal{D}$ are set to 1 with probability q. All events are independent. We require $q \geq 1/k$ so that r is a proper probability. Note that for any $g \in \mathcal{G}$ the probability that g is mutated is the same since for $d \in \mathcal{D}$, $\frac{1}{k} + (1 - \frac{1}{k})r = q$.

Note that both models differ from a simple *binomial* model, where each entry of A is mutated independently with a fixed probability. Since we condition on each patient having at least one mutation in $\mathcal{D}$, the entries of A corresponding to genes in $\mathcal{D}$ are not independent. In what follows, we let $\Gamma(g) = \{i : A_{ig} = 1\}$ denote the set of patients in which a gene g is mutated. Similarly, for a set M of genes, let $\Gamma(M)$ denote the set of patients in which at least one of the genes in M is mutated: $\Gamma(M) = \cup_{g \in M} \Gamma(g)$.

3 Finding Recurrently Mutated Genes

The standard approach to identify the driver genes is to identify recurrently mutated genes, i.e. those genes whose observed frequency of mutations is significantly higher than the expected *passenger mutation probability*[5,11,12]. This approach assumes a prior knowledge or a good estimate of the passenger mutation probability, the parameter q in our models. This approach is combined with a multi-hypothesis test to identify a list of genes, each mutated in significantly more patients than expected. The pseudocode for such a test is given in Algorithm RMG (Figure 1). (In Algorithm RMG we use Bonferroni correction for multiple hypothesis testing. Other corrections, like Benjamini-Hochberg [1] to control the *False Discovery Rate*, are possible. The results of this section also apply to those other corrections.)

We first analyze the D>P model of Section 2. We start by showing that if q is known and the number of patients is sufficiently large, then Algorithm RMG outputs all the driver genes with high probability.

Algorithm. RMG

```
Input: An $m \times n$ mutation matrix $A$, a probability $q$ that a gene contains a
       passenger mutation in a patient, a significance level $\alpha$.
Output: Set $\mathcal{O}$ of recurrently mutated genes.
1 $\mathcal{O} \leftarrow \emptyset$;
2 for $g \in \mathcal{G}$ do
3     $\Gamma(g) \leftarrow \{i : A_{ig} = 1\}$;
4     $p_g \leftarrow \Pr[B(m, q) \geq |\Gamma(g)|]$;
5     if $p_g \leq \frac{\alpha}{n}$ then $\mathcal{O} \leftarrow \mathcal{O} \cup \{g\}$;
6 return $\mathcal{O}$;
```

Fig. 1. Pseudocode of the algorithm for finding recurrently mutated genes, based on a single-gene test

Theorem 1. *Suppose an $m \times n$ mutation matrix A is generated by the the* D>P *model, the family wise error rate of the test is $\alpha = \frac{1}{2n^{\varepsilon}}$ and Algorithm* RMG *outputs $\mathcal{O}$. If $m \geq \frac{2k^2(1+\varepsilon)}{(1-q)^2} \ln 2n$ for a constant $\varepsilon > 0$, then* $\Pr[\mathcal{O} \neq \mathcal{D}] \leq \frac{1}{n^{\varepsilon}}$.

Proof. The p-value calculations and the Bonferroni correction in Algorithm RMG guarantee that the probability that any gene $g \notin \mathcal{D}$ is included in the output set $\mathcal{O}$ is bounded by $\alpha = \frac{1}{2n^{\varepsilon}}$. It remains to prove that if $m \geq \frac{2k^2(1+\varepsilon)}{(1-q)^2} \ln 2n$ the probability that any $d \in \mathcal{D}$ is not included in $\mathcal{O}$ is bounded by $\frac{1}{2n^{\varepsilon}}$.

Consider a gene $d \in \mathcal{D}$. Let $X_i = 1$ if gene d is mutated in patient i, and $X_i = 0$ otherwise. Note that for $i \neq j$, X_i and X_j are independent. Let X be the number of patients in which d is mutated. We have $X = \sum_{i=1}^{m} X_i$. To compute $\mathbf{E}[X_i]$ we observe that a driver gene is mutated with probability 1 when it contains the driver mutation, and with probability q otherwise. Since the gene d containing the driver mutation is chosen uniformly at random among all the k genes in $\mathcal{D}$, we have $\mathbf{E}[X_i] = \frac{1}{k} + \left(1 - \frac{1}{k}\right) q$. Thus $\mathbf{E}[X] = \sum_{i=1}^{m} \mathbf{E}[X_i] = m(\frac{1}{k} + \left(1 - \frac{1}{k}\right) q) > mq$. Let $t = \frac{1}{k}\left(\frac{1-q}{2}\right)$. By the Chernoff-Hoeffding bound:

$$\Pr[X \leq \mathbf{E}[X] - tm] = \Pr[X \leq m\mathbf{E}[X_i] - tm] \leq e^{-\frac{2m^2t^2}{m}} \leq \frac{1}{2n^{1+\varepsilon}}.$$

Since $|\mathcal{D}| < n$, by union bound we have:

$$\Pr[\exists d \in \mathcal{D} \text{ mutated in } \leq (\mathbf{E}[X] - tm) \text{ patients}] \leq n\frac{1}{2n^{1+\varepsilon}} = \frac{1}{2n^{\varepsilon}}.$$

Thus with probability at least $1 - \frac{1}{2n^{\varepsilon}}$ all genes in $\mathcal{D}$ are mutated in at least $\mathbf{E}[X] - tm$ patients. Let $B(m, q)$ be a binomial random variable with parameters m, q. Using the Chernoff-Hoeffding bound we can upper bound the p-value p_d that Algorithm RMG derives for $d \in D$:

$$p_d \leq \Pr[|B(m, q) - mq| \geq tm] \leq e^{-2\frac{t^2m^2}{m}} \leq \frac{1}{2n^{1+\varepsilon}}.$$

Thus, with probability at least $1 - \frac{1}{2n^{\varepsilon}}$ for any $d \in \mathcal{D}$ the number of patients with a mutation in d is such that its p-value satisfies $p_d < \alpha/n$ and thus it is included in the output set $\mathcal{O}$. □

Theorem 1 shows that in the D>P model an estimate of the passenger mutation probability q and a sufficient number of patients are enough to identify the driver genes. This is not the case in the D=P model. It is easy to see that in D=P model the expected number of rows in which a column g is mutated is the same for all $g \in \mathcal{G}$, that is for all $g \in \mathcal{G}$ we have $\mathbf{E}[|\Gamma(g)|] = qm$. In fact, the number $|\Gamma(d)|$ of patients in which a gene $d \in \mathcal{D}$ is mutated and the number $|\Gamma(g)|$ of patients in which gene $g \notin \mathcal{D}$ is mutated are both binomial random variables $B(m, q)$. We thus have the following.

Fact 1. *Under the D=P model, the probability distribution of $|\Gamma(d)|$ for $d \in \mathcal{D}$ and $|\Gamma(g)|$ for $g \notin \mathcal{D}$ are the same. Thus Algorithm RMG cannot identify the genes in $\mathcal{D}$ for any number of patients m.*

4 A Weight Function to Identify Driver Pathways

In this section we analyze a method that identifies the set $\mathcal{D}$ of driver genes with no prior information on the passenger mutation probability q, and works for both the D>P and D=P models. The method relies on a weight function $W(M)$, defined on sets of genes, first introduced in [14]. The measure W quantifies the extent to which a set simultaneously exhibits both: (i) high *coverage*: most patients have at least one mutation in the set; (ii) high *exclusivity*: nearly all patients have no more than one mutation in the set. (For lack of space, some proofs of the results in this section are omitted. They will be included in the full version of this work.)

For a set of genes, M, we define the coverage overlap $\omega(M) = \sum_{g \in M} |\Gamma(g)| - |\Gamma(M)|$. Note that $\omega(M) \geq 0$, with equality if and only if the mutations in M are mutually exclusive. To account for both the coverage, $\Gamma(M)$, and the coverage overlap, $\omega(M)$, we define the weight function of M:

$$W(M) = |\Gamma(M)| - \omega(M) = 2|\Gamma(M)| - \sum_{g \in M} |\Gamma(g)|.$$

Finding a set M of genes with maximum weight is in general a computationally challenging problem (it is NP-hard in the worst case). Nonetheless, we showed in [14] that under some assumptions on the distribution of mutations in patients, a greedy algorithm will identify the maximum weight set. We also proposed a Markov Chain Monte Carlo (MCMC) approach that samples sets of genes with probability proportional to their weight.

Based on the coverage and exclusivity properties of a driver pathway we expect it has the highest weight among all sets of size k. In this section we formalize this intuition for our generative models and show that under the two models the maximum weight set is easy to compute. We use M_k^* to denote the set of size k with maximum weight (M_k^* may not be unique).

We start with the D>P model. Note that the parameter q controls the expected number of passenger mutations in a set of k passenger genes. Since passenger mutations are relatively rare and k (the number of genes in a driver pathway) is relatively small, we expect that a set of k passenger genes will not have a mutation in the majority of the patients. Thus we assume that the probability $1-(1-q)^k$ that a set of k passenger genes contains a mutation is less than a constant $a < \frac{1}{2}$. Since $1-(1-q)^k \approx qk$ we have $q \leq \frac{a}{k}$. For ease of exposition in what follows we use $a = \frac{1}{4}$, so that $q \leq \frac{1}{4k}$.

Let $M_{k,\ell} \subset \mathcal{G}$ be a set of k genes with exactly ℓ genes of $\mathcal{D}$, that is $M_{k,\ell} = \{d_1, d_2, \ldots, d_\ell\} \cup \{g_1, \ldots, g_{k-\ell}\}$ with $d_j \in \mathcal{D}$ for $1 \leq j \leq \ell$, and $g_j \in \mathcal{G} \setminus \mathcal{D}$ for $1 \leq j \leq k-\ell$. We first prove that $\mathbf{E}[W(M_{k,\ell})]$ is monotone in ℓ.

Lemma 1. *Let $q \leq \frac{1}{4k}$. For $0 \leq \ell \leq k-1$: $\mathbf{E}[W(M_{k,\ell+1})] \geq \mathbf{E}[W(M_{k,\ell})] + \frac{m}{2k}$.*

Next we show that for sufficiently large number of patients m, the random value $W(M_{k,\ell})$ is concentrated near its expectation.

Theorem 2. *Suppose A is generated by the D>P model with $q \leq \frac{1}{4k}$. For $m \geq 8k^3(k+\varepsilon)\ln n$, and for $0 \leq \ell \leq k-1$, $Pr[\exists M_{k,\ell}$ s.t. $|W(M_{k,\ell}) - \mathbf{E}[W(M_{k,\ell})| \geq \frac{m}{4k}] \leq \frac{1}{n^\varepsilon}$.*

Combining the results of Lemma 1 and Theorem 2 we have

Corollary 1. *If $m \geq 8k^3(k+\varepsilon)\ln n$, then $\Pr[M_k^* \neq \mathcal{D}] \leq \frac{1}{n^\varepsilon}$.*

Corollary 1 shows that with sufficient number of patients the set $\mathcal{D}$ can be identified by finding the set of maximum weight, without an estimate of the probability q that a gene is mutated as a passenger. It was shown in [14] that with an arbitrary mutation distribution identifying the set of maximum weight is NP-Hard. However, a simple corollary of Theorem 2 shows that in our generative model computing a set of maximum weight is easy.

Corollary 2. *If $m \geq 8k^3(k+\varepsilon)\ln n$ and $q \leq \frac{1}{4k}$, a greedy algorithm that computes the weight function of up to $O(nk)$ sets finds M_k^* with failure probability $\leq \frac{1}{n^\varepsilon}$.*

Proof. Start with an arbitrary set M of k genes. Now consider the elements of $M = \{g_1, \ldots, g_k\}$ one after the other in a greedy process: for $g_j \in M$, find $w = \arg\max_{g \in G \setminus M} W(M \setminus \{g_j\} \cup \{g\})$. If $W(M) < W(M \setminus \{g_j\} \cup \{w\})$, substitute g_j with w in M; then move to g_{j+1}. Theorem 2 guarantees that if w is inserted in M, it is in $\mathcal{D}$, and that when a gene $g_j \in M \setminus \mathcal{D}$ is considered, it will be switched with a gene $d \in \mathcal{D} \setminus M$. □

We now consider the D=P model. Analogously to what we proved under the D>P model, we prove that maximizing the weight function W identifies the driver pathway $\mathcal{D}$ when mutation data from enough patients is available.

Theorem 3. *Suppose A is generated by the D=P model. If $m \geq \frac{k^3(k+\varepsilon)}{2(1-q)^{2k+2}}\left(\frac{k-1}{k}\right)^{2k}\ln n$, then $\Pr[M_k^* \neq \mathcal{D}] \leq \frac{1}{n^\varepsilon}$.*

We prove that a simple greedy algorithm, similar to the one proposed for the D>P model, identifies the set M_k^* of maximum weight under the D=P model.

Corollary 3. *If $m \geq \frac{k^3(k+\varepsilon)}{2(1-q)^{2k+2}} \left(\frac{k-1}{k}\right)^{2k} \ln n$, a greedy algorithm that computes the weight function of up to $O(n^2)$ sets finds M_k^* with failure probability $\leq \frac{1}{n^\varepsilon}$.*

Thus under the D=P model we identify the driver pathway $\mathcal{D}$ by maximizing $W(M)$. Recall that Algorithm RMG cannot find driver genes under this model (Section 3, Fact 1). Also note that when $q \leq 1/2$ and the probability $(1-q)^k$ that a set of k genes in $\mathcal{G} \setminus \mathcal{D}$ is not mutated in a patient is greater than $\frac{1}{2}\left(\frac{k-1}{k}\right)^k$ (this occurs when passenger mutations are relatively rare, for example when $q \approx 1/k$) the bound on m in Corollary 3 is the same as the bound in Corollary 2. That is, the weight W identifies the set $\mathcal{D}$ under both models with the same number of patients.

For completeness, we also analyze the Monte-Carlo Markov Chain approach proposed in [14] to sample sets of genes with distribution exponentially proportional to their weight. The states of the Markov chain are the subsets of $\mathcal{G}$ of size k. If $M^{(t)}$ is the state at time t, $M^{(t+1)}$ is computed choosing uniformly at random a gene $w \in \mathcal{G}$ and a gene $v \in M^{(t)}$, and setting $M^{(t+1)} = M^{(t)} \setminus \{v\} \cup \{w\}$ with probability $\min[1, e^{cW(M^{(t)}\setminus\{v\}\cup\{w\}) - cW(M^{(t)})}]$, and $M^{(t+1)} = M^{(t)}$ otherwise. It is easy to verify that the chain is ergodic with a unique stationary distribution $\pi(M) = \frac{e^{cW(M)}}{\sum_{R \in \mathcal{M}_k} e^{cW(R)}}$, where $\mathcal{M}_k = \{M \subset \mathcal{G} || M| = k\}$. The efficiency of this algorithm depends on the speed of convergence of the Markov chain to its stationary distribution.

In [14], we show that there is a non-trivial interval of values for c for which the chain is rapidly mixing without assuming any generative model for the mutation matrix. The analysis of [14] applied to D>P and D=P models requires $c < 1/k$. However, applying Lemma 1 and 2 under the D>P model, and Theorem 3 under the D=P model we show that for any $c > 0$ the process rapidly converges to the set $\mathcal{D}$.

Theorem 4. *Suppose that A is generated by the D>P model with $q \leq \frac{1}{4k}$, or the D=P model with $q \leq 1/2$ and $(1-q)^k \geq \frac{1}{2}\left(\frac{k-1}{k}\right)^k$. For $m \geq 8k^3(k+\varepsilon)\ln n$ and any $c > 0$, the MCMC converges to the set $\mathcal{D}$ in $O(nk \log k)$ iterations with probability $\geq 1 - \frac{1}{n^\varepsilon}$.*

5 Experimental Results

In this section we compare the single gene test provided in Algorithm RMG and the weight function $W(M)$ to detect the set of driver genes using mutation data simulated using the D>P model. In particular, we use the greedy algorithm of Section 4 (see Corollary 2) to identify the set M_k^* of maximum weight, where $k = |\mathcal{D}|$.

We generated mutation data according to the D>P model with $k = |\mathcal{D}| = 20$, $q = 0.0125$, $n = 10000$. We set $\alpha = 0.005$ for Algorithm RMG which corresponds

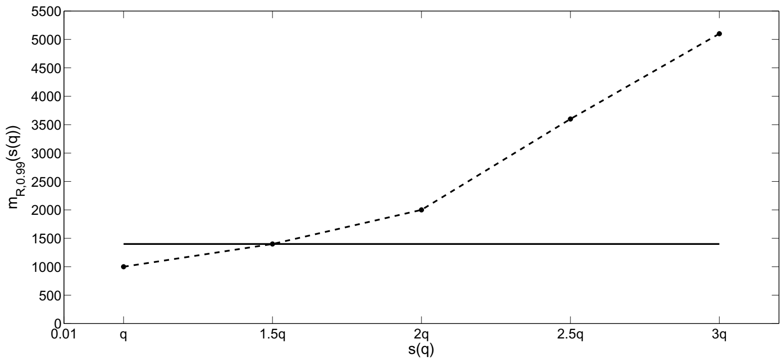


Fig. 2. Number of patients $m_{R,0.99}(s(q))$ required to identify the driver pathway $\mathcal{D}$ with Algorithm RMG , for different estimates $s(q)$ of the probability q (dashed). Number of patients $m_{G,0.99}$ required to identify $\mathcal{D}$ with the greedy algorithm (solid).

to $\varepsilon = 0.5$. To compare the performance of the two approaches, we measured the minimum number of patients required to detect the driver pathway $\mathcal{D}$ over a range of estimates of the passenger mutation probability q. Specifically, let $E_{s(q)}$ = "estimate $s(q)$ of q is used by Algorithm RMG". Let $m_{R,x}(s(q)) = \min_m\{\Pr[\mathcal{O} = \mathcal{D}|E_{s(q)}] > x\}$ be the minimum number of patients required for Algorithm RMG to output $\mathcal{O} = \mathcal{D}$ with probability $> x$ over all $m \times n$ mutation matrices generated by the model when the estimate $s(q)$ is used. Similarly, let $\mathcal{P}$ be the output of the greedy algorithm of Corollary 2. Let $m_{G,x} = \min_m\{\Pr[\mathcal{P} = \mathcal{D}] > x\}$ be the minimum number of patients required for the greedy algorithm to output $\mathcal{D}$ with probability $> x$ over all $m \times n$ mutation matrices generated by the model. Recall that $m_{G,x}$ does not depend on $s(q)$ by Corollary 2. Figure 2 shows the values of $m_{R,0.99}(s(q))$ and $m_{G,0.99}$ as a function of $s(q)$. We varied $s(q)$ starting from $s(q) = q$ (i.e., q is perfectly estimated) and gradually increased $s(q)$ while maintaining $s(q) < 1/k$. The latter condition assures that $s(q)$ is strictly smaller than the expected probability of mutation of any gene in $\mathcal{D}$, a necessary condition for Algorithm RMG to be able to identify $\mathcal{D}$. To estimate $m_{R,0.99}$ and $m_{G,0.99}$ we generated 100 mutation matrices for each $m_i = i * 100$ patients for $1 \leq i \leq 52$. Figure 2 shows that $m_{R,0.99}(s(q))$ is monotonically increasing with $s(q)$. When the estimate of q is perfect, the greedy algorithm requires more patients than Algorithm RMG to correctly identify the set $\mathcal{D}$, but when the estimate $s(q)$ is larger than the true value of q, $m_{R,0.99}(s(q))$ increases and becomes much larger than $m_{G,0.99}$. (Typically, an overestimate of q is used so that the test for recurrent genes in conservative [10]). Note that even when $s(q) = q$, $m_{G,0.99}$ is close to $m_{R,0.99}(q)$, while the bounds in Theorem 1 and Corollary 2 give $\frac{m_{G,0.99}}{m_{R,0.99}(s(q))} \geq 1000$. Similar results were obtained when comparing $m_{R,0.95}(s(q))$ and $m_{G,0.95}$; i.e. the minimum number of patients for which Algorithm RMG and the greedy algorithm report the driver set $\mathcal{D}$ at least 95% of the time (data not shown).

Finally, we consider the case $s(q) < q$ where the estimate of q is smaller than its true value. In this case, some genes not in $\mathcal{D}$ (false positives) are eventually reported by Algorithm RMG. For example, with $s(q) = 0.8q$ and $m = 1000$ (for which the correct result is always reported when $s(q) = q$), Algorithm RMG reports false positives in approximately 16% of the datasets.

6 Conclusions

We investigate the problem of detecting recurrently mutated genes and pathways using two simple generative models of driver mutations in cancer. In the first D>P model, where the driver mutation probability is larger than the passenger mutation probability, we prove a bound on the number of patients required to detect all driver genes with high probability using a single gene test of recurrence. In the second D=P model, where the driver mutation probability and passenger mutation probability cannot be distinguished, it is impossible to identify driver genes using the single gene test for *any* number of patients. We prove that under either model, the weight function that we defined in [14] is maximized by a driver pathway. Thus, with mutation data from enough patients, it is possible to identify driver pathways *without* an estimate of the passenger mutation probability q. In particular, we show that a simple greedy algorithm finds driver pathways with high probability. We also show that an MCMC approach converges rapidly. Finally, we present results on simulated data showing that the greedy algorithm successfully identifies the driver pathway with fewer patients than the single gene test when the estimate of q deviates from its real value.

In practice, any test that identifies driver genes by recurrent mutations requires a good estimate of q. An underestimate of q leads to false positive predictions of driver genes, while an over estimate (i.e. a conservative estimate to minimize false positives) increases the number of patients required to find driver genes. The passenger mutation probability is derived from the background mutation rate (BMR), which is difficult to measure as it depends on a number of parameters whose values are not easily determined. There has been extensive discussion in the community about appropriate ways to estimate the BMR and find recurrently mutated genes [7,11]. Therefore, methods that do not require an estimate of the BMR, as the ones we provide here, can give increased power in the discovery of driver genes. However, further study of more sophisticated mutation models is necessary. For example, we assume a constant passenger mutation probability q across all genes, but models that allow q to vary by gene would be useful in applications and warrant further investigation.

Acknowledgements. We thank the anonymous reviewers for helpful suggestions that improved the manuscript. This work is supported by NSF grants IIS-1016648 and CCF-1023160, the Department of Defense Breast Cancer Research Program, the Alfred P. Sloan Foundation, the Susan G. Komen Foundation and in part by the MIUR of Italy under project AlgoDEEP prot. 2008TFBWL4. BJR is also supported by a Career Award at the Scientific Interface from the Burroughs Wellcome Fund.

References

1. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate. J. Royal Statistical Society 57, 289–300 (1995)
2. Boca, S.M., Kinzler, K.W., Velculescu, V.E., Vogelstein, B., Parmigiani, G.: Patient-oriented gene set analysis for cancer mutation data. Genome Biol. 11, R112 (2010)
3. Cerami, E., Demir, E., Schultz, N., Taylor, B.S., Sander, C.: Automated network analysis identifies core pathways in glioblastoma. PLoS ONE 5, e8918 (2010)
4. Deguchi, K., Gilliland, D.G.: Cooperativity between mutations in tyrosine kinases and in hematopoietic transcription factors in AML. Leukemia 16, 740–744 (2002)
5. Ding, L., et al.: Somatic mutations affect key pathways in lung adenocarcinoma. Nature 455, 1069–1075 (2008)
6. Efroni, S., Ben-Hamo, R., Edmonson, M., Greenblum, S., Schaefer, C.F., Buetow, K.H.: Detecting cancer gene networks characterized by recurrent genomic alterations in a population. PLoS ONE 6, e14437 (2011)
7. Getz, G., Hofling, H., Mesirov, J.P., Golub, T.R., Meyerson, M., Tibshirani, R., Lander, E.S.: Comment on The consensus coding sequences of human breast and colorectal cancers. Science 317, 1500 (2007)
8. Hahn, W.C., Weinberg, R.A.: Modelling the molecular circuitry of cancer. Nat. Rev. Cancer 2, 331–341 (2002)
9. McCormick, F.: Signalling networks that cause cancer. Trends Cell Biol. 9, M53–M56 (1999)
10. Parmigiani, G., et al.: Response to Comments on The consensus coding sequences of human breast and colorectal cancers. Science 317(5844), 1500 (2007)
11. Sjoblom, T., et al.: The consensus coding sequences of human breast and colorectal cancers. Science 314, 268–274 (2006)
12. The Cancer Genome Atlas Research Network. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. Nature 455(7216) 1061–1068 (2008)
13. Vandin, F., Upfal, E., Raphael, B.J.: Algorithms for detecting significantly mutated pathways in cancer. J. Comput. Biol. 18, 507–522 (2011)
14. Vandin, F., Upfal, E., Raphael, B.J.: *De novo* Discovery of Mutated Driver Pathways in Cancer. Genome Research (in press, 2011)
15. Varela, I., et al.: Exome sequencing identifies frequent mutation of the SWI/SNF complex gene PBRM1 in renal carcinoma. Nature 469, 539–542 (2011)
16. Vogelstein, B., Kinzler, K.W.: Cancer genes and the pathways they control. Nat. Med. 10, 789–799 (2004)
17. Yeang, C.H., McCormick, F., Levine, A.: Combinatorial patterns of somatic gene mutations in cancer. The FASEB Journal (2008)

Clustering with Overlap for Genetic Interaction Networks via Local Search Optimization

Joseph Whitney, Judice Koh, Michael Costanzo, Grant Brown, Charles Boone, and Michael Brudno

University of Toronto
jwhitney@cs.toronto.edu

Abstract. Algorithms for detection of modules in genetics interaction networks, while often identifying new models of functional modular organization between genes, have been limited to the output of disjoint, non-overlapping modules, while natural overlapping modules have been observed in biological networks. We present CLOVER, an algorithm for clustering weighted networks into overlapping clusters. We apply this algorithm to the correlation network obtained from a large-scale genetic interaction network of *Saccharomyces cerevisia*e derived from Synthetic Genetic Arrays (SGA) that covers ~4,500 non-essential genes. We compare CLOVER to previous clustering methods, and demonstrate that genes assigned by our method to multiple clusters known to link distinct biological processes.

Keywords: graph clustering, genetic interactions, local search.

1 Introduction

Recent developments in Synthetic Genetic Arrays have enabled the mapping of quantitative genetic interactions on a genome-wide scale [1]. A central computational task in the analysis of the genetic interaction networks is to cluster genes into coherent modules. The resulting modules provide a higher-level organization of the cell, and may be mined to make new predictions about gene functions and provide insights into cellular pathways. Such unsupervised organization of a network's nodes into highly-interconnected modules is termed *graph clustering*.

Several general-purpose graph clustering methods that use a number of diverse algorithmic approaches have been applied to genetic and protein interaction networks. [2][3][4][5][6][7]The RNSC algorithm[4] is based on stochastic local search. It was developed for clustering un-weighted general graphs and has been applied to the problem of predicting protein complexes in physical networks. MCL [5] is a general-purpose algorithm for clustering weighted graphs, and has been widely used to analyze both physical and genetic interaction networks. MCL uses stochastic matrix operations to simulate flow through a weighted network, and determines clusters as groups of nodes connected by a large number of high-weight paths. Graph summarization [6] is a paradigm for clustering graphs in which the output is a coarse-grained *summary* of the input graph, with super-nodes representing groups of nodes with similar interaction patterns and super-edges represent groups of interactions between members of super-nodes. The *cost* of this summary can be formulated in terms of the number of single-edge additions or deletions necessary to recover the input graphs. For module detection in genetic interactions specifically, the network of genetic *profile correlations*, rather than individual genetic interactions, has been found useful for determining cohesive modules even via hierarchical clustering [8][9]. In particular Ulitsky *et al.* achieved better results using clustering models biased toward strong profile correlation within modules versus strong intra-modular "monochromatic" alleviating interactions[10], despite the known enrichment of such interactions in protein complexes. All of these methods partition a network into non-overlapping clusters. However, as many genes participate in multiple biological processes, it has been recognized that a more general approach, allowing clusters to *overlap* (so that multiple genes may participate in any number of distinct clusters), would facilitate the identification of clusters of genes which are biologically more meaningful, and possibly exhibit higher functional coherence.

The need for clustering algorithms that explicitly model overlap was first proposed by Palla *et al.* [11]. Their algorithm for identifying overlapping clusters, CFINDER, is based on identifying overlapping k-connected subgraphs between maximal cliques, using a simple combinatorial definition of cluster which applies only to unweighted networks. The algorithm has exponential running time on dense graphs, making it impractical for analysis of genetic interaction networks, which are weighted, dense graphs. More recently Wang *et al.* proposed HACO[7], an algorithm that extends average-linkage hierarchical clustering to allow clusters to be re-used in multiple agglomerative iterations, producing a lattice which can be used to induce overlapping clusters which are similar in cohesion. It cannot be used to find overlaps between clusters of widely differing cohesion due to an exponential increase in the number of clusters which must be considered.

Some pre-existing methods designed to find modules with particular characteristics operate by repeatedly seeding a search from a single node (as in MCODE[2]) or interaction (as in the within/between-pathway model of Kelley and Ideker [12], using a similar search algorithm to Sharan et al[13]). Implicit overlaps may exist in the cumulative output and highly-overlapping modules are identified using a threshold of overlap with a higher-scoring module – overlaps are viewed as redundancies and not a targeted feature of optimization. Finally, the MCL algorithm can occasionally produce overlapping clusters around exact symmetries; on the real data used in this study no such overlaps were observed.

In this paper we present CLOVER (CLustering with OVERlap), a novel method for clustering weighted networks into overlapping clusters. CLOVER combines the ability to deal with large, weighted networks via local search optimization. We have applied our method to two large-scale genetic interaction networks [1][8] and CLOVER was able to identify many clear instances of overlapping modularity in this dataset. We demonstrate that the modules identified by CLOVER show significant enrichment for known functional annotations, on par with previously published results, while the overlaps between clusters identify genes which indeed link distinct biological processes.

2 Methods

CLOVER is a local search optimization algorithm. Like all local search algorithms, CLOVER optimizes a cost function – an evaluation of the quality of a solution – by perturbing an intermediate solution via local moves. In this section we first present the cost function utilized in the CLOVER algorithm, which captures objective criteria for good overlapping clusters. We then describe the local search method used to optimize the cost function on an input network.

2.1 CLOVER Cost Function

The CLOVER cost function represents a trade-off between two desiderata of a clustering

- *cohesion,* represented by many, or high-weight edges within each cluster, and
- *separation*, represented by few, or low-weight edges joining vertices assigned to different clusters

while accommodating *overlap* between clusters. The intuition behind the cost function may be best understood by considering the case of a simple, unweighted graph. We *penalize* (assign a cost to) any pair of nodes that violates the cohesion or separation desiderata. For each pair of vertices u, v that are connected by an edge, but lie in different clusters, a *cross edge* penalty is charged. Similarly, for each pair of vertices u, w lying in the same cluster, but not connected by an edge a *missing edge* penalty is charged (see Figure 1 (a)). By combining these penalties over all pairs of nodes, we have a standard clustering cost function for unweighted graphs: the "cost" of a clustering is the number of edges crossing between clusters, and the number of edges "missing" from within clusters (see, e.g. [4]).

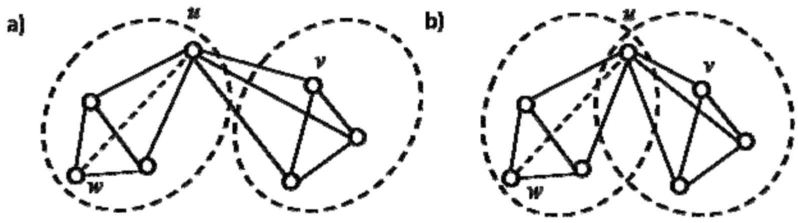


Fig. 1. a) in the non-overlapping case, there are two deviations from a perfect clustering: the edge $(\boldsymbol{u},\boldsymbol{v})$ is a "cross edge" between vertices in different clusters while $(\boldsymbol{u},\boldsymbol{w})$ is a "missing edge". b) $(\boldsymbol{u},\boldsymbol{v})$ is still a cross-edge from the perspective of $\boldsymbol{v}$: $\boldsymbol{u}$ is not a cross edge from the perspective of v: u is in the same cluster. But $(\boldsymbol{u},\boldsymbol{v})$ is a "half cross edge" from the perspective of $\boldsymbol{u}$, since it is in only one out of the two clusters of $\boldsymbol{u}$. Similarly $(\boldsymbol{u},\boldsymbol{w})$ is a "missing edge" from the perspective of $\boldsymbol{w}$, since $\boldsymbol{u}$ is in the same cluster, but is only a "half missing edge" from the perspective of $\boldsymbol{u}$.

The CLOVER cost function generalizes this simple cost function in two ways. First, we deal with weighted networks by applying a penalty to missing and cross edges proportionate to their weights. For simplicity we assume that edge weights are scaled to lie in the interval [0,1]. For all edges within clusters we apply a penalty that is proportionate to 1 minus the weight: a cluster-internal edge with zero weight is assigned the "full" penalty while an edge with the "full" weight is assigned zero penalty. Conversely, for cross edges the penalty is directly proportional to the weight.

The second generalization deals with overlapping clusters, which requires modification of the cross-edge definition, as a single edge may be internal to one cluster, while crossing between several pairs of others. Our solution is to regard each penalty applying to an edge (u, v) as being calculated twice: once from the perspective of u and once from the perspective of v. Then for *each cluster* containing u, we apply a "cross-edge" penalty if v is not also in that cluster, and a "missing edge" penalty otherwise. The penalties are divided by the number of clusters containing u. Figure 1 (b) illustrates the application of this cost function to an example network.

The formal definition of the cost function is simplified by assuming a *complete* weighted graph where every pair of vertices u, v exists in E and is assigned a weight by ω, which may be 0. We further assume that while a clustering may assign any number of nodes to each cluster and vice versa, no node is allowed to be "unclustered" and thus not contribute to the clustering cost defined below. In practice, very small clusters (< 3 nodes) are pruned from the final results.

Formally, we define a *clustering* $\mathcal{C}$ of a weighted network $G = (V, E, \omega)$, where $\omega: E \mapsto \mathbb{R}$ assigns a weight in the range [0,1] to each edge, as a collection of sets $C \subset V$, or *clusters*. We use the notation C_v to denote the set of clusters to which a vertex v is assigned by $\mathcal{C}$.

The cost function is defined as:

$$f(G, \mathcal{C}) = \sum_{u \in V} (\alpha \times x(u, \mathcal{C}, G) + (1 - \alpha) \times m(u, \mathcal{C}, G))$$

where

$$x(u, \mathcal{C}, G) = \frac{1}{|C_u|} \sum_{C \in C_u} \sum_{v \in V - C} \omega(u, v)$$

represents the total weight of edges (u, v) leading "out of" a cluster from vertex u (cross-edge cost), and

$$m(u, \mathcal{C}, G) = \tfrac{1}{|C_u|} \textstyle\sum_{C \in C_u} \sum_{v \in C} (1 - \omega(u, v))$$

represents the total weight "missing" from each cluster containing u (missing edge cost). The real parameter α represents a trade-off between the two factors x and m, across all nodes.

The cost function calculates, for each vertex u and each cluster C, a penalty for each edge (u, v). If $v \in C$, the penalty is equal to $(1 - \alpha)(1 - \omega(u, v))$. This rewards higher-weight edges within the same cluster. If $v \notin C$, the penalty is just $\alpha \times \omega(u, v)$, penalizing higher-weight edges leading out of C from u. In the absence of overlap, the same penalty will be applied symmetrically to v, in effect doubling all penalties. In the case of overlapping clusters, this cost is calculated *for each* cluster containing u – some of these clusters may include v, and some not. The sum of these cluster-wise penalties is then divided by the number of clusters to which the node u is assigned. An example of how this cost function promotes overlap is illustrated in Figure 2. Because both the x and m penalty factors are scaled by the number of clusters in which each vertex appears, increasing or decreasing the number of clusters to which a vertex is assigned does not increase its contribution to the cost.

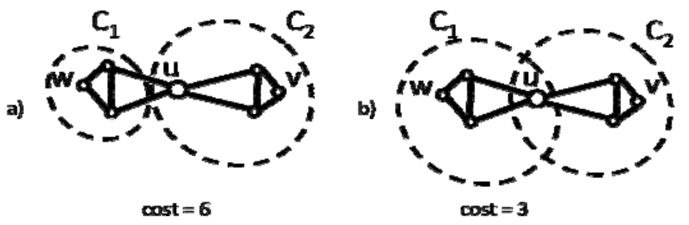


Fig. 2. Example of how the cost function promotes overlap. Edges drawn are assumed to have weight 1, edges not drawn are assumed to have weight 0. In the scenario at left, the total cost of this small clustering is 6: $\mathbf{1}$ for each of the two edges shown leaving $\boldsymbol{C_2}$ from vertex u, 1 for each of the (same) two edges leaving $\boldsymbol{C_1}$, 1 for the "missing" edge from $\boldsymbol{u}$ to $\boldsymbol{v}$, and 1 for the (same) "missing" edge from $\boldsymbol{v}$ to $\boldsymbol{u}$. In the scenario at right, the total cost is reduced to 3: $(\mathbf{2}+\mathbf{2})/\mathbf{2}$ for each edge leaving a cluster from $\boldsymbol{u}$, and $(\mathbf{1}+\mathbf{1})/\mathbf{2}$ for each "missing" edge from $\boldsymbol{u}$ to a neighbour in a shared cluster.

2.2 Local Search Algorithm

The CLOVER algorithm is an instance of the *local search* method, a type of discrete search which is typically applied to computationally difficult (e.g. NP-hard) combinatorial problems. Its goal is to minimize the value of an objective *cost function* that quantifies the fitness of a clustering as a real number, such that a "perfect" clustering has a cost of 0.

Like all local search algorithms CLOVER defines moves between *candidate solutions* – in this case, clusterings of an input network. Starting with the trivial clustering in which each vertex is assigned to one unique cluster, a new solution is constructed by applying a move, or transition operation, to the current one. There are three types of moves. The first is a *toggle* (reversal) of the inclusion of some particular vertex in some particular cluster. For a given current solution $\mathcal{C}$ for input network G, the set of possible toggle move transitions can be defined as $V \times \mathcal{C}$. The second type is an *all-in*: given an ordered pair of clusters (C, D), assign to D every vertex in C which is not already so assigned. The third type is an *all-out*: given clusters (C, D), remove from D every vertex in C which is currently assigned to D. CLOVER chooses the transition which will result in a new solution with lowest cost, b evaluating the cost of all clusterings that can be obtained by one transition from the current solution.

Any optimization search algorithm needs some mechanism for hill-climbing – that is, of moving the search away from local minima. To this end CLOVER follows a variant of *tabu search*. A list of recently moved nodes, the *tabu list*, is maintained, and nodes on the tabu list are not moved again until a pre-defined number of moves τ has passed. We have used a τ value of 50, having determined experimentally that larger values up to 150 slow convergence to a solution without obtaining a better-scoring result.

Inherent to a local search algorithm is the need to determine a stopping condition, since it is impossible to know whether the best possible solution has been found. In some cases the stopping condition may simply be a bounded number of transitions or total running time. Either of these could be used with CLOVER, but for the results reported in this study we have used a convergence criterion: the algorithm will continue to iterate through solutions until some sufficiently large number of transitions has been made without encountering a better (lower-cost) solution than the best recorded so far. The convergence interval should be large relative to the tabu list length, so that the algorithm may return repeatedly to moves previously marked tabu. We used a convergence interval of 500 moves in all cases reported here. We discarded clusters containing fewer than three nodes – these nodes may still be reported in other, larger clusters.

The performance of CLOVER for a particular application depends greatly on the value of the α parameter. This parameter allows one to trade-off the size of the clusters and their connectivity: a lower α results in clusters which are smaller and more strongly connected, while a higher α results in clusters which are larger and more weakly connected. In Figure 3 (a) we plot the GO enrichment precision, recall, and F-measure of all the clusterings obtained with different values of α for the Costanzo et al [1] data. For comparison, we also plot the same metric for MCL with different values of the inflation parameter *I* in Figure 3 (b). Because the F-measure combines precision and recall into a single value, it is not as sensitive to cluster granularity. For this particular data set, the best results are achieved over a relatively wide plateau corresponding to α settings 0.55 - 0.75.

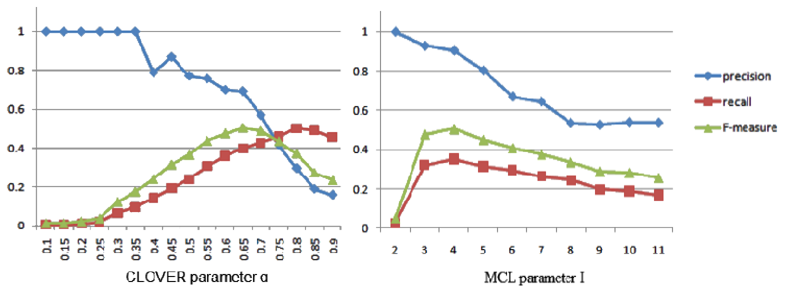


Fig. 3. Precision, recall and F-measure of GO enrichment for clustering obtained from CLOVER and MCL

3 Results

In this section we describe the application of CLOVER to the correlation networks of two high-throughput genetic interaction data sets, demonstrating CLOVER's ability to identify functionally cohesive modules while identifying genes working at the interface of these functions.

3.1 Genetic Interaction Data Sets

Two genetic interaction data sets for *S. cerevisiae* were used in this study. The more recent and larger SGA (*Synthetic Genetic Array*) data set is obtained from [1] and comprises more than 5.4 million interactions involving 4443 genes. The raw genetic interactions from that work are modeled after the Fisher definition of epistasis as quantitative deviations from the expected multiplicative combination of independently functioning genes [14], which defines genetic interaction strength as follows:

$$\epsilon = f_{ab} - f_a f_b$$

f_a and f_b denote the single mutant fitness of gene *a* and gene *b* respectively, and f_{ab} is the fitness of the double-knockout mutant of genes *a* and *b*. The Collins *et al.* EMAP (*Epistatic Mini-Array Profile)* data set [9] is an earlier yeast double-knockout array, including 743 genes. While there is some overlap between the genes/interactions in these studies, they are independent and use different statistical techniques to compute interaction scores.

Both data sets consist of a matrix of scores, with each row corresponding to a query gene, and each column to one single-knockout library gene with which each query was crossed. A positive score indicates greater fitness in the resulting double mutant than expected according to a model based on single-knockout fitness measures for each of the two genes individually; such interactions are denoted *alleviating*. A negative score indicates lower fitness and is denoted *aggravating* [1]. In both data sets some individual scores are missing from the matrix.

Because the individual interaction scores are noisy and incomplete, we use the *correlation network* of the original interaction data, following the lead of previous studies (e.g. [15][9]). Each edge of the network corresponds to the similarity in the interaction profile between the two genes, obtained from the interactions by computing the Pearson correlation of each pair of genes' interaction profiles across all other genes.

Positions in the profile where either interaction value was missing were ignored. To obtain a complete graph in which all edge weights fall in the interval [0,1], negative correlations were truncated to zero values, followed by scaling the remaining values linearly so that the highest correlation value is mapped to 1. We found this produced superior results to the two obvious alternatives: taking the absolute value of the correlation, and scaling all values into the range [0,1].

3.2 Evaluation on Genetic Interactions Networks

To validate the efficacy of CLOVER for the analysis of genetic interaction data, we clustered the correlation networks generated from quantitative interaction data from [9] and [1] respectively, as described above. We also obtained clusters using MCL for the same networks, over a range of the "inflation" parameter *I* which determines the granularity of the derived clusters. Additionally, a recent study [10] evaluated a number of other clustering techniques on the data from[9], and we compare our results to the best reported result achieved in that study by using genetic interactions alone.

3.3 Evaluation of Clusters

To evaluate the fitness of clusters identified using CLOVER, we tested the enrichment of clusters for known groups of functionally related genes. For this purpose we used all biological process annotations from the Gene Ontology (GO[16]). We considered a cluster to be "matched" to an annotation or complex if its enrichment was significant (P-value≤ 0.05) according to a Bonferroni-corrected hypergeometric test, following similar procedures in recent studies[10][6].

The number of matches based on this criterion, divided by the total number of clusters, gives a *precision* value P between 0 and 1 for each clustering compared to each benchmark. The number of matched benchmark groups, divided by the total number of groups, gives a *recall* value R. These values are combined into the harmonic mean *F-measure* defined as $F = P \times R/(P + R)$.

Table 1. Cluster Enrichment Summary for E-Map Network vs GO Annotations

Method	Precision (P)	Recall (R)	F-measure (P*R)/(P+R)	#clusters	#matches	#overlapping match pairs	#nodes in two clusters
Ulitsky et al	0.58	0.31	0.41	48	28	N/A	N/A
MCL $I = 4$	1.0	0.20	0.33	9	9	N/A	N/A
CLOVER $\alpha = 0.45$	0.72	0.32	0.44	53	38	Total: 12 Matched one-sided: 2 two-sided: 6	138

Table 2. Cluster Enrichment Summary for SGA Network vs GO Annotations

Method	Precision (P)	Recall (R)	F-measure (P*R)/(P+R)	#clusters	#matches	#overlapping match pairs	#nodes in two clusters
MCL $I = 4$	0.9	0.35	0.51	42	38	N/A	N/A
CLOVER $\alpha = 0.65$	0.69	0.4	0.51	189	131	Total: 40 Matched one-sided: 3 two-sided: 8	138

We compared the performance of CLOVER to MCL, a leading general-purpose clustering algorithm, on correlation networks derived from the Collins et al. EMAP and Costanzo et al. SGA data sets. We chose MCL because of its favourable performance relative to other methods for clustering of the EMAP data set[10]. For both algorithms we obtained the clusterings over a wide range of cluster granularity parameters (I for MCL and α for CLOVER), and present the best of these results. The performance of CLOVER over the full range of α parameters is described in Results. For the Collins EMAP dataset we also compare our results with the clustering obtained by the "Correlation" method of Ulitsky *et al.*[10], provided by the authors. Details of the best clustering obtained with each method, as determined by the corresponding F-measure, are presented in Tables 1 and 2. These results show that the CLOVER clustering captures co-annotated genes as well as MCL on the SGA correlation network, and better than both MCL and the "Correlation" method on the EMAP correlation network, with a significantly larger number of clusters in each case. Tables 1 and 2 additionally list the number of *matched overlaps* in each CLOVER clustering, as defined in the following section. The number of genes assigned to two clusters is also given; note that at the given parameter setting no nodes were assigned to more than two clusters.

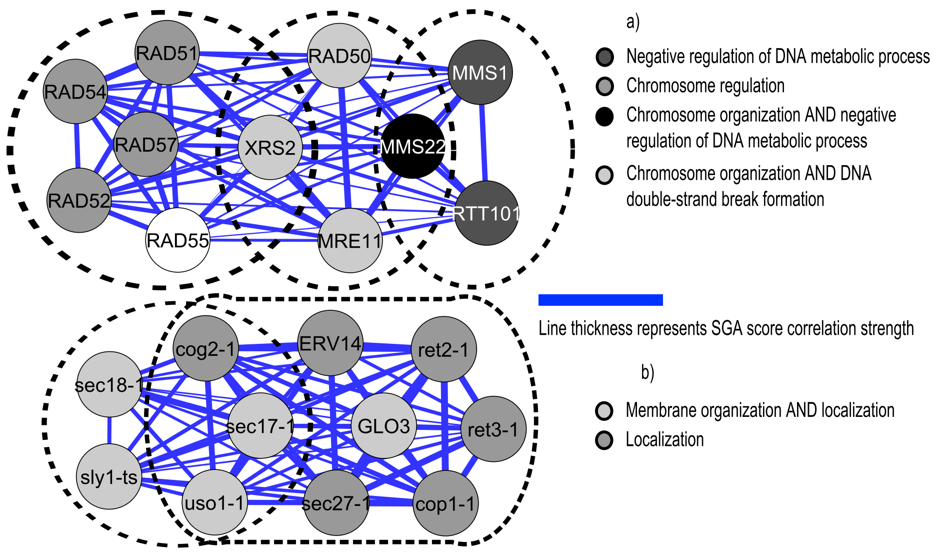

RAD51
RAD50
MMS1
RAD54
RAD57
XRS2
MMS22
RAD52
RTT101
RAD55
MRE11
a)
Negative regulation of DNA metabolic process
Chromosome regulation
Chromosome organization AND negative regulation of DNA metabolic process
Chromosome organization AND DNA double-strand break formation
Line thickness represents SGA score correlation strength
cog2-1
ERV14
ret2-1
sec18-1
sec17-1
GLO3
ret3-1
sly1-ts
uso1-1
sec27-1
cop1-1
b)
Membrane organization AND localization
Localization

Fig. 4. Examples of double-sided matching pairs of clusters found by GO annotation mining. a) The genes *XRS2* and *MMS22* both participate in two of the three clusters shown. b) An example of three genes participating in two clusters.

3.4 Evaluation of Overlaps

We further assessed pairs of overlapping clusters with respect to the benchmarks according to the following criteria. A *two-sided matched overlap* is a pair of clusters C, D such that

- Both C and D are enriched for *distinct* biological processes (e.g. they are both "true positives")
- The overlapping region between C and D (i.e. the set of genes assigned to both clusters) contains genes which are annotated for both processes

Similarly, a *one-sided matched overlap* is a pair of clusters C, D such that

- Both C and D are enriched for *distinct* biological processes
- *The overlapping region betwee*n C and D (i.e. the set of genes assigned to both clusters) contains genes which are annotated for at least one of these processes

In addition to these categories we also counted the number of simple overlapping matches – pairs of clusters, each matching some biological process, which overlap. The number of matched overlaps of each kind is included in tables 1 & 2.

In the remainder of this section, all specific examples of overlapping clusters are taken from the clustering of the SGA correlation network [1], using the parameter $\alpha = 0.65$. Figure 4(a) illustrates two two-sided overlaps. Each of the three clusters is enriched for a distinct subset of the GO biological process categories "negative regulation of DNA metabolic process", "chromosome organization", and "DNA double-strand break formation". Both of the genes *XRS2* and *MMS2* are assigned to multiple clusters, and share the annotations for which those clusters are enriched. Figure 4(b) illustrates two clusters which share 3 overlapped genes. While all genes in both clusters are annotated with the broad annotation "localization", the cluster depicted at left is also enriched for "membrane organization". Two of the three genes shared by both clusters, *sec17-1* and *uso1-1*, are annotated with both biological process categories.

While this validation of overlaps participating in enriched biological functions is encouraging, the definitions of GO biological process annotations are coarse-grained, and cannot fully capture the fine distinctions evidenced by genetic interaction profiles and their modularity. Some genes are clearly and strongly connected to multiple clusters, and this can be validated by detailed analysis of the individual genes in each cluster. Figure 6 illustrates the intersection of two such clusters, with the *CTF4* gene assigned to both. One cluster includes genes involved in sister chromatid cohesion, while the other contains genes involved in DNA replication. In particular, the proteins encoded by *CTF18*,*CTF8*, and *DCC1* form a part of the replication factor C-like complex required for sister chromatid cohesion [18][19], while *MRC1* & *CSM3* gene products have been demonstrated present at replication forks [20][21]. POL32[22][23], *RAD27*[24], and *ELG1* [25][26][27] are all involved in lagging strand DNA synthesis. The *CTF4* gene product has also been found to act at replication forks[28][29][30][31], and in sister chromatin cohesion [19][32][18].

4 Future Improvements

Local search provides a great deal of control and flexibility due to the use of an explicit objective function. Recent work on special models of modularity in genetic interaction networks (e.g. [33][17]) suggest that more complex and domain-specific scoring schemes can be used to model specific types of modularity; we are interested in incorporating these models, possibly in combination, explicitly into our algorithm.

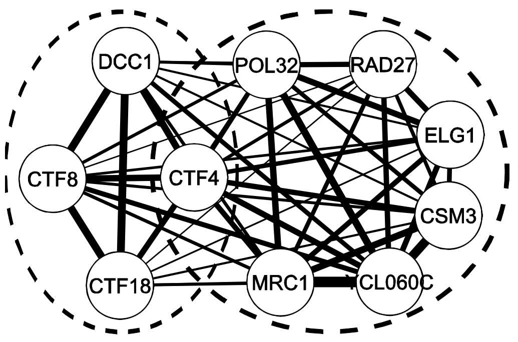


Fig. 5. Clusters overlapping at *CTF4*. The cluster depicted at left contains genes involved in sister chromatid cohesion. The cluster depicted at right contains genes involved in DNA replication.

This flexibility comes at a cost in computational complexity; currently our implementation takes roughly 6 hours to produce the reported results on our Dell Xeon-based servers, compared to e.g. under 20 minutes for MCL which does not perform explicit optimization. A common approach to this kind of intractability is to use a faster, unspecialized (and possibly non-overlapping) clustering method as a pre-processing step to reach a clustering which is "nearer" to optimal than a random or trivial starting point. Speeding up the algorithm in this way will allow faster exploration of different parameters and objective function variations.

References

1. Costanzo, M., Baryshnikova, A., Bellay, J., Kim, Y., Spear, E., Sevier, C., Ding, H., Koh, J., Toufighi, K., Mostafavi, S., Prinz, J., Onge, R., VanderSluis, B., Alizadeh, S., Bahr, S., Brost, R., Chen, Y., Cokol, M., Deshpande, R., Li, Z., Li, Z.-Y., Lian, W., Marback, M., Paw, J., San Luis, B.-J., Shuteriqi, E., Tong, A., van Dyk, N., Wallace, I., Whitney, J., Weirauch, M., Zhong, G., Zhu, H., Houry, W., Brudno, M., Ragibizadeh, S., Papp, B., Roth, F., Giaever, G., Nislow, C., Troyanskaya, O., Bussey, H., Bader, G., Gingras, A., Morris, Q., Kim, P., Kaiser, C., Myers, C., Andrews, B., Boone, C.: The Genetic Landscape of a Cell. Science 327(5964), 425–431 (2010)
2. Bader, G., Hogue, C.: An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics 4(2) (January 2003)
3. Hartuv, E., Shamir, R.: A Clustering Algorithm based on Graph Connectivity. Information Processing Letters 76, 175–181 (2000)
4. King, A., Przulj, N., Jurisica, I.: Protein complex prediction via cost-based clustering. Bioinformatics 20(17), 3013–3020 (2004)
5. van Dongen, S.: A Clustering Algorithm for Graphs. Technical Report 1386- 3681, Centrum voor Wiskunde en Informatica, Amsterdam (2000)
6. Navlakha, S., Schatz, M., Kingsford, C.: Revealing Biological Modules via Graph Summarization. Journal of Computational Biology 16(2), 253–264 (2009)
7. Wang, H., Kakaradob, B., Karotki, L., Fiedler, D., Shales, M., Shokat, K., Walther, T., Krogan, N., Koller, D.: A Complex-Based Reconstruction of the S. cerevisiae Interactome. Molecular and Cellular Proteomics 8(6), 1361–1381 (2009)
8. Tong, A.: Global Mapping of the Yeast Genetic Interaction Network. Science 303(5659), 808–813 (2004)
9. Collins, S., Miller, K., Maas, N., Roguev, A., Fillingham, J., Chu, C., Schuldiner, M., Gebbia, M., Recht, J., Shales, M., Ding, H., Xu, H., Cheng, B., Andrews, B., IngVarsdottir, K., Han, J., Boone, C., Berger, S., Hieter, P., Zhang, Z., Emili, A., Alic, C., Toczyski, D.P., Weissmann, J.: Functional dissection of protein complexes involved in yeast chromosome biology using a genetic interaction map. Nature 446, 806–810 (2007)
10. Ulitsky, I., Shlomi, T., Kupiec, M., Shamir, R.: From E-MAPs to module maps: dissecting quantitative genetic interactions using physical interactions. Molecular Systems Biology 4(209) (May 2008)
11. Palla, G.D.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814–818 (2005)
12. Kelley, R., Ideker, T.: Systematic interpretation of genetic interactions using protein networks. Nature Biotechnology 23(5), 561–566 (2005)

13. Sharan, R., Ideker, T., Kelley, R., Shamir, R., Karp, R.: Identification of Protein Complexes by Comparative Analysis of Yeast and Bacterial Protein Interaction Data. J. Comp. Bio. 12(6), 835–836 (2005)
14. Fisher, R.: The correlations between relatives on the supposition of Mendelian inheritence. Trans. R. Soc. 52, 399–433 (1918)
15. Schuldiner, M., Collins, S., Thompson, N., Denic, V., Bhamidipati, A., Punna, T., Ihmels, J., Andrews, B., Boone, C., Greenblatt, J., Weissman, J., Krogan, N.: Exploration of the function and organization of the yeast early secretory pathway through an epistatic miniarray profile. Cell 123, 507–519 (2005)
16. Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S.: Gene ontology: a tool for the unification of biology. Nature Genetics 25, 25–29 (2000)
17. Mayer, M., Gygi, S., Aebersold, R., Hieter, P.: Identification of RFC (Ctf18p, Ctf8p, Dcc1p): an alternative RFC complex required for sister chromatid cohesion in S. cerevisiae. Mol. Cell (May 2001)
18. Hanna, J., Kroll, E.S.: Saccharomyces cerevisiae CTF18 and CTF4 are required for sister chromatid cohesion. Molecular Cell Biology (May 2001)
19. Bando, M., Katou, Y., Komata, M., Tanaka, H., Itoh, T., Sutani, T., Shirahige, K.: Csm3, Tof1, and Mrc1 form a heterotrimeric mediator complex that associates with DNA replication forks. J. Biol. Chem. (October 2009)
20. Szyjka, S., Viggiani, C., Aparicio, O.: Mrc1 is required for normal progression of replication forks throughout chromatin in S. cerevisiae. Mol. Cell 19(5), 691–697 (2005)
21. Johansson, E., Majka, J., Burgers, P.: Structure of DNA polymerase delta from Saccharomyces cerevisiae. J Biol. Chem. 276(47), 43824–43828 (2001)
22. Stith, C., Sterling, J., Resnick, M., Gordenin, D., Burgers, P.: Flexibility of eukaryotic Okazaki fragment maturation through regulated strand displacement synthesis. J.Biol. Chem. 283(49), 34129–34140 (2008)
23. Liu, Y., Kao, H., Bambara, R.: Flap endonuclease 1: a central component of DNA metabolism. Annu. Rev. Biochem. 73, 589–615 (2004)
24. Bellaoui, M., Chang, M., Ou, J., Xu, H., Boone, C., Brown, G.: Elg1 forms an alternative RFC complex important for DNA replication and genome integrity. EMBO J 22(16), 4304–4313 (2003)
25. Kanellis, P., Agyei, R., Durocher, D.: Elg1 forms an alternative PCNAinteracting RFC complex required to maintain genome stability. Curr. Biol. 13(18), 1583–1595 (2003)
26. Ben-Aroya, S., Koren, A., Liefshitz, B., Steinlauf, R., Kupiec, M.: ELG1, a yeast gene required for genome stability, forms a complex related to replication factor C. Proc. Natl. Acad. Sci. 100(17), 9906–9911 (2003)
27. Lengronne, A., McIntyre, J., Katou, Y., Kanoh, Y., Hopfner, K., Shirahige, K., Uhlmann, F.: Establishment of sister chromatid cohesion at the S. cerevisiae replication fork. Mol. Cell (September 2006)
28. Gambus, A., van Deursen, F., Polychronopoulos, D., Foltman, M., Jones, R., Edmondson, R., Calzada, A., Labib, K.: A key role for Ctf4 in coupling the MCM2-7 helicase to DNA polymerase alpha within the eukaryotic replisome. EMBO J 28(19), 2992–3004 (2009)
29. Tanaka, H., Katou, Y., Yagura, M., Saitoh, K., Itoh, T., Araki, H., Bando, M., Shirahige, K.: Ctf4 coordinates the progression of helicase and DNA polymerase alph. Genes Cells 14(7), 807–820 (2009)
30. Gambus, A., Jones, R., Sanchez-Diaz, A., Kanemaki, M., van Deursen, F., Edmondson, R., Labib, K.: GINS maintains association of Cdc45 with MCM in replisome progression complexes at eukaryotic DNA replication forks. Nat. Cell Biol. 8(4), 358–366 (2006)

31. Warren, C., Eckley, D., Lee, M., Hanna, J., Hughes, A., Peyser, B., Jie, C., Irizarry, R., Spencer, F.: S-phase checkpoint genes safeguard high-fidelity sister chromatid cohesion. Mol. Biol. Cell 15(4), 1724–1735 (2004)
32. Segre, D., DeLuna, A., Church, G.M., Kishony, R.: Modular Epistasis in Yeast Metabolism. Nature Genetics 37(1), 77–83 (2005)
33. Ulitsky, I., Shamir, R.: Pathway redundancy and protein essentiality revealed in the Saccharomyces cerevisiae interaction networks. Mol. Syst. Biol. 3(104) (April 2007)
34. Brohee, S., van Helden, J.: Evaluation of clustering algorithms for proteinprotein interaction networks. BMC Bioinformatics 7(488) (November 2006)
35. Mewes, H., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schüller, C., Stocker, S., Weil, B.: MIPS: a database for genomes and protein sequences. Nucleic Acids Res. 28(1), 37–40 (2000)

Dynamic Programming Algorithms for Efficiently Computing Cosegmentations between Biological Images

Hang Xiao[1,⋆], Melvin Zhang[2,⋆], Axel Mosig[3,1,⋆], and Hon Wai Leong[2,1,⋆]

[1] Department of Biophysics, CAS-MPG Partner Institute and Key Laboratory for Computational Biology, 320 Yue Yang Road, 200031 Shanghai, China
[2] Department of Computer Science, National University of Singapore, Computing 1, 13 Computing Drive, Singapore 117417, Republic of Singapore
[3] Bioinformatics Group, Department of Biophysics, Ruhr University Bochum, Universitätsstraße 150, 44801 Bochum

Abstract. Algorithms for comparing trees have recently been found relevant in the context of bioimage analysis. While previously proposed algorithms deal with problems that are computationally hard in general, we propose efficient algorithms for restricted versions that are able to handle significantly larger instances in practice. We propose two dynamic programming algorithms for the so-called tree assignment problem, which generalizes bipartite matchings to trees. We formulate restricted versions that are tractable by a dynamic programming algorithm. Furthermore, we describe a second dynamic programming algorithm that deals with the efficient computation of certain weights between so-called component trees that can be applied to obtain certain cosegmentations in bioimaging applications. Our investigations indicate that our algorithms are both efficient and effective, supported by evaluating the influence of the restrictions imposed by the dynamic programming formulation on a collection of image data.

1 Introduction

Algorithms for comparing trees have been well-studied due to their relevance in areas such as comparison of RNA secondary structures. However, their use in image processing was established only recently. Here, comparing hierarchical image representations obtained from the so-called component tree naturally leads to a tree comparison problem that generalizes bipartite matching [10,15] instead of edit distance, which is used for comparing RNA secondary structures [13,4] and other applications.

In the most general form, generalizations of bipartite matching to trees have recently been shown to be $\mathcal{NP}$-hard [2]. Instances encountered in practice with few dozens of vertices are often solvable within seconds using integer linear programming (ILP), as shown by two of the authors in [10,15]. Yet, the computational

⋆ These authors contributed equally.

hardness imposes constraints on the size of the instances that are considered in practice, so that constraint versions tractable in polynomial time are of high relevance for practical applications. In this paper, we introduce a restricted version of the problem that is solved in polynomial time using a dynamic programming formulation. The same restriction was earlier used in the context of tree edit distance, thus our results extend the class of tree alignments and distance measures surveyed in [1].

Currently, the main application of tree assignments is in bioimaging, where such weighted assignments are computed between component trees that represent fluorescence-based images [15]. Introduced to image processing by Jones [6], the component tree represents all connected components that occur by setting some threshold in a gray-scale image. As these connected components are hierarchically ordered, they define the component tree, which is computed in near-linear time using a union-find data structure [11]. In [15], the weights between vertices of two component trees is the Jaccard index between the areas represented by the vertices. While naïve approaches require at least quadratic time to compute the weights between all pairs of vertices, we here introduce a dynamic programming algorithm that is optimal in the sense that its running time is proportional to the number of pixels in the image plus the number of non-zero overlap weights. In an approach similar to the cosegmentations introduced in [15], Mattes et al. [9] also used the idea of computing assignments between vertices of component trees. They adopt a top-down approach whose exact constraints on the matching results remain uncharacterized.

In [15], cosegmentations obtained from tree assignments are used in the context of cell tracking, requiring to solve a sequence of $N-1$ cosegmentation instances for a microscopic video sequence of N time frames. In order to evaluate whether the restriction to the three-point condition imposed by our dynamic programming approach still yields results useful in practice, we evaluate our approach using the same data as in [15]. Interestingly, there is also theoretical evidence that the three-point condition does not restrict meaningful results as long as background inhomogeneity – a common phenomenon in fluorescence microscopy – is similar among the cosegmented images.

2 Problem Formulation

General tree assignments. Let S and T denote two rooted unordered trees, with vertices U and V, respectively. We will be dealing with *tree assignments* between two trees, which are sets $M = \{(u_1, v_1), \dots, (u_k, v_k)\} \subset U \times V$ such that for any two distinct indices $1 \leq i, j \leq k$, neither u_i is an ancestor/descendant of u_j nor v_i is an ancestor/descendant of v_j. We refer to the set of all possible assignments between S and T as $\operatorname{match}(S, T)$. Given a weighting function $w \colon U \times V \to \mathbb{R}_{\geq 0}$ that assigns a score $w(u, v)$ whenever u is matched with v, we can assign a weight $W(M) := \sum_{(u,v)\in M} w_{u,v}$ to an assignment M. Putting things together, this allows us to define the *tree assignment problem*, which is to find the maximum weighted tree assignment, given S, T and w. The tree assignment problem is a generalization of the maximum weighted bipartite matching problem.

Constrained tree assignments. The tree assignment problem can be modelled and solved using integer linear programming [15]. In general, however, the tree assignment problem has recently been shown to be $\mathcal{NP}$-hard [2]. As such, we consider a constrained tree assignment inspired by the constrained tree edit distance [16]. Similar to the tree edit distance, we found that the constrained version of the tree assignment problem is solvable in polynomial time. We call the constraint the *three-point condition* because it introduces a restriction on the topology of trees with three leaves. We adapt the recursions for the constrained tree edit distance to solve the restricted tree assignment problem.

The three-point condition involves the lowest common ancestor of two vertices a, b in a tree, which we denote by $\mathrm{lca}(a, b)$. Now, we call $M \in \mathrm{match}(S, T)$ a *constrained tree assignment* if for any three assignments $(a_1, a_2), (b_1, b_2), (c_1, c_2) \in M$, we have

$$\mathrm{lca}(a_1, b_1) = \mathrm{lca}(a_1, c_1) \iff \mathrm{lca}(a_2, b_2) = \mathrm{lca}(a_2, c_2)$$

Intuitively, the three-point condition ensures that for any three pairs of vertices in an assignment, the topology of the two induced subtrees are identical. This implies that distinct subtrees in one tree are assigned to distinct subtrees in the other tree.

As a short hand notation, $\mathrm{cmatch}(S, T)$ will refer to the set of all constrained tree assignments between S and T. Corresponding to the tree assignment problem, the *constrained tree assignment problem* is to find the maximum weighted constrained tree assignment between two trees based on a weight function w.

Component trees. In our image analysis setting, we are dealing trees S and T that are component trees. These are computed from images $I\colon P \to \{0, \ldots, q-1\}$, where P is the set of all image coordinates and q the number of gray values. For a threshold $0 \leq \theta < q$, the pixel set $\{p \in P \mid I(p) \geq \theta\}$ falls apart into connected components (e.g. with regard to the 4-neighborhood or 8-neighborhood of pixels in a 2D image). The connected components for all possible thresholds $0, \ldots, q-1$ are hierarchically ordered, defining the component tree in which each vertex represents one connected component. In a component tree, each vertex v is associated with a (connected) set of pixels, we denote this set by $\beta(v)$.

There is an intuitive interpretation of constrained tree assignments between component trees involving background inhomogeneities that are of high relevance in microscopic images [7]. This interpretation suggests that the restriction to constrained tree assignments does not affect results as long as background inhomogeneity (which can be dealt with using rolling-ball-type algorithms [14]) is consistent among the two underlying images, as illustrated in Fig. 1.

Calculating overlap weights. An important part of obtaining cosegmentation is to compute the vertex weights between two given component trees S and T. To simplify notation, we use the same symbol β for the vertex-to-pixels mappings of S and T, so that for vertex u in S, $\beta(u)$ refers to a pixel set in the image underlying S and for vertex v in T refers to a pixel set in T's image. A weighting function to be used for cosegmentation, as introduced in [15], is the Jaccard index:

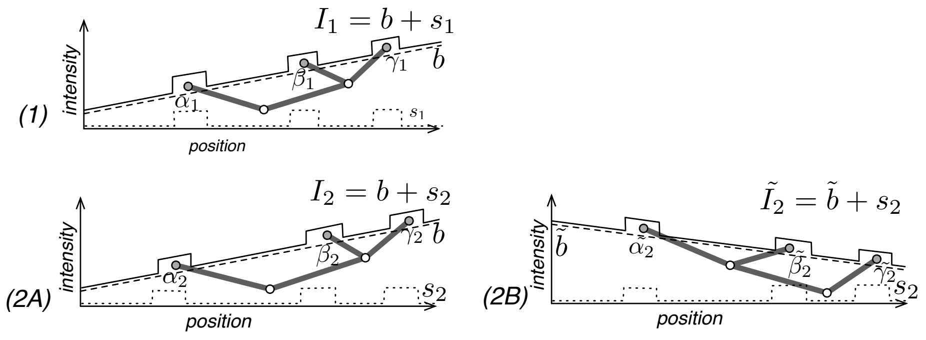


Fig. 1. *Relationship between the three-point condition and background inhomogeneity in images: (1)* Often, a biological image such as I_1 contains an inhomogeneous background b in addition to the actual (e.g. fluorescence) signal s_1, which contains three objects that are represented by vertices $\alpha_1, \beta_1, \gamma_1$ in the corresponding component tree. *(2A)* Given a second images $I_2 = b + s_2$ with same background b and similar objects represented by vertices $\alpha_2, \beta_2, \gamma_2$, a tree assignment involving the three assignments $(\alpha_1, \alpha_2), (\beta_1, \beta_2), (\gamma_1, \gamma_2)$ respects the three-point condition. *(2B)* If, however, we have image $\tilde{I}_2$ with same signal s_2 but completely different background $\tilde{b}$, an assignment involving the three assignments $(\alpha_1, \tilde{\alpha_2}), (\beta_1, \tilde{\beta_2}), (\gamma_1, \tilde{\gamma_2})$ will violate the three-point condition.

$$w(u, v) = |\beta(u) \cap \beta(v)| / |\beta(u) \cup \beta(v)|$$

Before solving either the constrained or the unconstrained tree assignment problem, it is necessary to compute the weights between all pairs of vertices.

3 A Quasi-Linear Time Algorithm for Computing Overlap Weights

Before computing tree assignments between two trees, the weight between each pair of nodes, one in each component tree, needs to be computed. In practice, calculating weights is a major bottleneck for computing cosegmentations. Here, we propose a linear time dynamic programming algorithm for calculating overlap weights by systematically utilizing the inclusion relationship of a node and its children in a component tree.

A component tree T can be described as $T = (V, E, \alpha, \beta)$, where V is the set of vertices in the tree T, and α and β are two mappings from vertices to sets of pixels in an image. For each vertex v, $\beta(v)$ is the connected area in the image that is associated with v, while $\alpha(v)$ is the connected area of v *excluding* the connect areas of the *children of* v. The pixels in $\alpha(v)$ belong *exclusively* to v and not to any of its children. Furthermore, let v_i denote the i-th child of vertex v and $C(v)$ be the set of children of v. Then, we have

$$\beta(v_i) \cap \beta(v_j) = \emptyset \quad \text{for all } v_i \neq v_j \in C(v)$$

Hence, $\beta(v)$ can be decomposed as follows:

$$\beta(v) = \alpha(v) \cup \bigcup_{v_i \in C(v)} \beta(v_i)$$

That is to say, $\beta(v)$ can be *partitioned* into $\alpha(v)$ and $\beta(v_i)$, $v_i \in C(v)$. This will be the guiding principle of our dynamic programing algorithm for weight calculation.

Let $S = (V_S, E_S, \alpha, \beta)$ be the first component tree, and $T = (V_T, E_T, \alpha, \beta)$ be the second component tree. Our task is to compute the Jaccard index between each vertex u in S and vertex v in T, defined as

$$w(u, v) = |\beta(u) \cap \beta(v)| \ / \ |\beta(u) \cup \beta(v)|$$

We can rewrite the above as

$$w(u, v) = \frac{|\beta(u) \cap \beta(v)|}{|\beta(u)| + |\beta(v)| - |\beta(u) \cap \beta(v)|}$$

Thus, the weight calculation between vertex u in S and vertex v in T can be rephrased as calculating intersections between $\beta(u)$ and $\beta(v)$. As a shortcut notation, we denote the cardinality of the intersection of $\beta(u)$ and $\beta(v)$ as

$$\beta\beta(u, v) = |\beta(u) \cap \beta(v)|$$

Similarly, we define

$$\alpha\beta(u, v) = |\alpha(u) \cap \beta(v)|$$

$$\alpha\alpha(u, v) = |\alpha(u) \cap \alpha(v)|$$

Decomposing $\beta(v)$ into $\alpha(v)$ and $\beta(v_i)$, we can split $\beta\beta(u, v)$ into

$$\beta\beta(u, v) = \alpha\beta(u, v) + \sum_{u_i \in C(u)} \beta\beta(u_i, v),$$

while $\alpha\beta(u, v)$ can be decomposed into

$$\alpha\beta(u, v) = \alpha\alpha(u, v) + \sum_{v_i \in C(v)} \alpha\beta(u, v_i),$$

For our dynamic programming algorithm, the idea is to use three dynamic programming tables $\alpha\alpha$, $\alpha\beta$, and $\beta\beta$. Based on the dependency relationship between them, we compute them in the following order:

$$\alpha\alpha(u, v) \rightarrow \alpha\beta(u, v) \rightarrow \beta\beta(u, v)$$

Let P be the set of all pixels in an image, and $T(V, E, \alpha, \beta)$ be its component tree. Then, $\alpha(v)$ is a partition of P

$$P = \bigcup_{v \in V} \alpha(v) \tag{1}$$

This allows us to define a reverse mapping of $\alpha^{-1} : P \to V$ that identifies for each pixel $p \in P$, the unique vertex v that satisfying $p \in \alpha(v)$. This reverse mapping allows us to calculate all $\alpha\alpha(u, v)$ in time $O(|P|)$.

Based on the above recurrence relations, both $\beta\beta(u, v)$ and $\alpha\beta(u, v)$ can be calculated in a dynamic programming fashion by postorder traversal of the component trees (see Algorithm 1). This leads to a time complexity of $O(|S| \cdot |T)$.

Worst-case time complexity. For the whole weight calculation process, we first compute all $\alpha\alpha$ weights, then the $\alpha\beta$ weights, and finally the $\beta\beta$ weights, yielding a total running time of $O(|P| + |S| \cdot |T|)$.

From Eqn. (1), we get

$$|V| = \frac{|P|}{\text{avg}_{v \in V}(|\alpha(v)|)}$$

We always have $|V| \leq |P|$, since $\text{avg}_{v \in V}(|\alpha(v)|) \geq 1$. For large images ($|P| \sim 10^6$) such as the ones considered in Section 5, the full component tree contains about 10^5 vertices. However, after pruning, the tree size decreases dramatically, to typically much less than 500, as observed in [15]. Usually, for large images, we have $\text{avg}_{v \in V}(|\alpha(v)|) > |V|$, so $|V|^2 \ll |P|$. Under these circumstances, the total running time, dominated by the number of pixels, is quasi-linear w.r.t $|P|$.

4 Dynamic Programming Algorithm for Constrained Tree Assignment

Constrained tree assignments can be considered to be a special case of the constrained edit distance [16], where assigning node u to node v is equivalent to changing the label of node u to the label of node v. The cost of changing node u to v is $w(u, v)$ and the rest of the operations have zero cost. In addition, once we changed node u to v, we do not have to consider the descendants of u and v anymore.

Hence, our dynamic programming algorithm is a simplification of the dynamic programming algorithm for computing constrained edit distance between unordered labeled trees [16].

Let T_u be the tree rooted at node u and F_u be the forest of the subtrees of u. As a slight abuse of notation, we refer to T_v as a subtree of another tree S if vertex v belongs to some tree S. $M(T_u, T_v)$ is the optimal assignment between the two trees T_u and T_v and $W(T_u, T_v)$ is the score of the optimal assignment. The following lemmas establish the recurrence relation for W.

Lemma 1.

$$W(T_u, T_v) = \max \begin{cases} W(u, T_v) \\ W(T_u, v) \\ W(F_u, F_v) \end{cases}$$

$$W(u, T_v) = \max \begin{cases} w(u, v) \\ \max_{y \in C(v)} w(u, T_y) \end{cases} \qquad W(T_u, v) \quad = \max \begin{cases} w(u, v) \\ \max_{x \in C(u)} w(T_x, v) \end{cases}$$

Algorithm 1. Computing all overlap weights between component trees S and T

```
Compute post-order enumerations of the vertices in $S$ and $T$ as $u_1, \dots, u_n$ and
$v_1, \dots, v_m$, respectively

{$\alpha\alpha$ weights calculation}
Initialize all $\alpha\alpha(u, v)$ to 0
**for** each pixel $p \in P$ **do**
   $u := \alpha_1^{-1}(p)$
   $v := \alpha_2^{-1}(p)$
   increase $\alpha\alpha(u, v)$ by one
**end for**

{$\alpha\beta$ weights calculation}
**for** $i$ from 1 to $n$ **do**
   **for** $j$ from 1 to $m$ **do**
      Initialize $\alpha\beta(u_i, v_j)$ to $\alpha\alpha(u_i, v_j)$
      **for** each child $c \in C(v_j)$ **do**
         $\alpha\beta(u_i, v_j) := \alpha\beta(u_u, v_j) + \alpha\beta(u_i, c)$
      **end for**
   **end for**
**end for**

{$\beta\beta$ weights and Jaccard index calculation}
**for** $i$ from 1 to $n$ **do**
   **for** $j$ from 1 to $m$ **do**
      Initialize $\beta\beta(u_i, v_j)$ to $\alpha\beta(u_i, v_j)$
      **for** each child $c \in C(u_i)$ **do**
         $\beta\beta(u_i, v_j) := \beta\beta(u_u, v_j) + \beta\beta(c, v_j)$
      **end for**
      {$|\beta(u)|$ and $|\beta(v)|$ are tracked during the building of the component trees}
      $w(u_i, v_j) := \beta\beta(u_i, v_j)/(|\beta(u_i)| + |\beta(v_j)| - \beta\beta(u_i, v_j))$
   **end for**
**end for**
```

Proof. Consider the nodes u and v, there are three possible cases: (1) $u \in M$, (2) $v \in M$, and (3) $u \notin M$ and $v \notin M$.

Case 1 ($u \in M$). Node u is matched to some node in T_v. In order to maximize the objective function node, u must be matched to some node x in T_v that maximizes $w(x, v)$.

Case 2 ($v \in M$). Similar to case 1.

Case 3 ($u \notin M$ and $v \notin M$). Since both u and v are not in M, we can remove them and find an optimal assignment between the remaining forests, F_u and F_v.

Algorithm 2. Computing the optimal assignment between two trees, S and T

```
Compute post-order enumerations of the vertices in $S$ and $T$ as $u_1, \dots, u_n$ and
$v_1, \dots, v_m$, respectively

{Matching a node $u_i$ in $S$ to all subtrees of $T$}
for $i$ from 1 to $n$ do
    for $j$ from 1 to $m$ do
        $W(u_i, T_{v_j}) := w(u_i, v_j)$
        for each child $c$ in $C(v_j)$ do
            $W(u_i, T_{v_j}) := \max(W(u_i, T_{v_j}), w(u_i, T_c))$
        end for
    end for
end for

{Matching a node $v_j$ in $T$ to all subtrees of $S$}
for $j$ from 1 to $m$ do
    for $i$ from 1 to $n$ do
        $W(T_{u_i}, v_j) = w(u_i, v_j)$
        for each child $c$ in $C(u_i)$ do
            $W(T_{u_i}, v_j) = \max(W(T_{u_i}, v_j), w(T_c, v_j))$
        end for
    end for
end for

{Matching a subtree of $S$ to a subtree of $T$}
for $i$ from 1 to $n$ do
    for $j$ from 1 to $m$ do
        Construct a weighted bipartite graph $G = (U \cup V, E)$, where $U = F_{u_i}$, $V = F_{v_j}$,
        and $E = \{(T_x, T_y, W(T_x, T_y)) \mid T_x \in F_{u_i} \text{and } T_y \in F_{v_j}\}$
        $M_G = \text{MaxWeightedBipartiteMatching}(G)$
        $W(T_{u_i}, T_{v_j}) = \max(W(u_i, T_{v_j}), W(T_{u_i}, v_j), \sum_{(T_x,T_y) \in M_G} W(T_x, T_y))$
    end for
end for
```

Lemma 2. *Let $\mathcal{M}$ be the set of all possible matchings between the trees in F_u and the trees in F_v. Then,*

$$W(F_u, F_v) = \max_{M \in \mathcal{M}} \sum\nolimits_{(T_x,T_y) \in M} W(T_x, T_y)$$

Proof. The key property of the three-point condition is that distinct trees in F_u is assigned to distinct trees in F_v, hence it suffices to consider one-to-one matchings between the trees in F_u and F_v. As we wish to maximize the final assignments, we maximize over all possible one-to-one matchings between trees in F_u and trees in F_v. This is precisely the *maximum weighted bipartite matching* problem where the two partite sets are the trees in F_u and F_v respectively and the weight between two trees is given by W.

We first compute $W(u, T_v)$ and $W(v, T_u)$ for all possible pair of nodes u and v, where u is in S and v is in T using dynamic programming. Then we compute $W(T_u, T_v)$ by solving a maximum weighted bipartite matching problem and combining that with the results from the previous step. The pseudocode for the whole algorithm is listed in Algorithm 2.

Worst-case time complexity. The number of subproblems in W is $O(|S| \cdot |T|)$ and except for the computation of the maximum weighted bipartite matching, each subproblem can be solved by taking the maximum of a fixed number of cases. Therefore, the bottleneck in this algorithm is in the computation of maximum weighted bipartite matching. The worst-case time complexity for computing the maximum weighted bipartite matching on a graph with n vertices and m edges is $O(n(m + n \lg n))$ [3], thus the worst case time complexity for computing the optimal assignment between F_u and F_v is $O((n_u + n_v)(n_u n_v + (n_u + n_v) \lg(n_u + n_v)))$, where n_u is the number of children of node u and n_v is the number of children of node v. Hence, the worst-case time complexity of our algorithm is

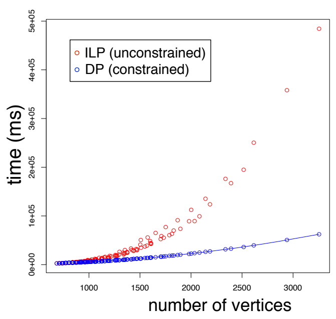
$$\begin{aligned}
&\sum_{u \in U} \sum_{v \in V} C \times (n_u + n_v)(n_u n_v + (n_u + n_v) \lg(n_u + n_v)) \\
&\leq C \sum_{u \in U} \sum_{v \in V} D \cdot (n_u n_v) + D \lg D \cdot (n_u + n_v) \\
&= C \left(D(|S| \cdot |T|) + D \lg D (|S| \cdot |T|) \right) \\
&\leq 2C \left((|S| \cdot |T|) D \lg D \right) \\
&= O((|S| \cdot |T|)(\deg(S)) + \deg(T)) \lg(\deg(S) + \deg(T)))
\end{aligned}$$

where $\deg(S)$ is the maximum degree of a node in S and $D = \deg(S) + \deg(T)$.

5 Results

In our application, we are interested in using a pair of fluorescence-based images, taken in two consecutive time frames, to identify regions that corresponds to cells. Our general approach is to first convert each image into a hierarchal representation (component tree) and then perform tree assignment to identify similar segments in the two images.

In this section, we evaluated our dynamic programming algorithm for solving constrained tree assignment against the ILP approach using synthetic and real images from [15]. Real images display in-vivo time-lapse recordings of zebrafish brain with a green fluorescent protein expressed specific to microglia (neural immune cells); synthetic images contain ellipsoid objects perturbed by noise and different types of background inhomogeneity. Both synthetic and real images are three-dimensional involving around $200 \times 200 \times 40$ voxels each. We compared the running times and the scores obtained by the constrained tree assignments against the unconstrained tree assignment to evaluate whether the constraint affect solutions that are relevant in practice. As the quasi-linear time algorithm for computing overlap weights has been implemented, although not described, in [15], we did not further compare its running time to a naïve approach.



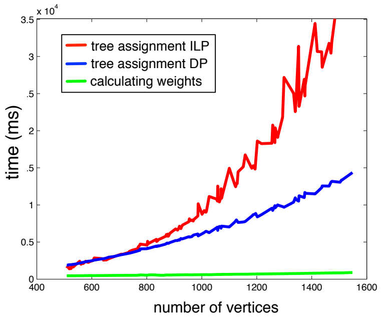


Fig. 2. *Running time of integer-linear programming approach vs. dynamic programming approach.* The number of vertices is the sum of the size of the two trees that were aligned. Instances of different size were derived from the microglia dataset from [15] by applying different pruning parameters for the component trees, again following [15]. The running time for computing weights becomes negligibly small using the quasi-linear time algorithm from Section 3 (right).

Running time of unconstrained versus constrained tree assignments. In the first experiment, we compared the running time of the integer linear programming approach from [15] with the dynamic programming approach for the constrained tree assignment as described in Section 4. As shown in Figure 2, our DP approach allows us to solve substantially larger instances within a few seconds compared

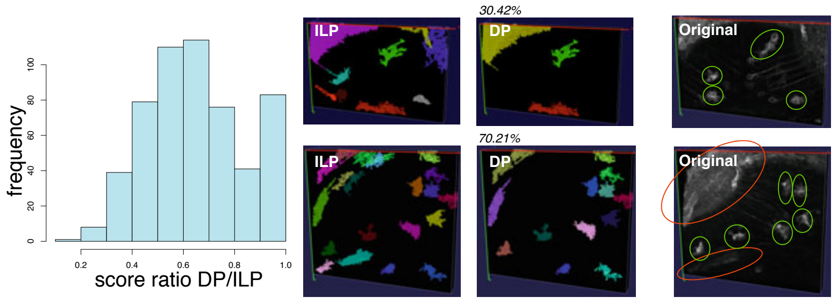


Fig. 3. *Left Part.* Histogram of score ratios between dynamic programming approach and integer linear programming approach The majority of instances achievea a score of more than 60% of the unconstrained version and can be expected to be reliable. *Right Part. (Top.)* Example of a dataset where the constrained version (middle) achieves a score of only 30.42% of the unconstrained version (left), thus missing relevant segments representing microglia indicated by green circles in the original image (right). *Bottom.* Instance where the constrained version achieves 70.21% of the unconstrained version's score. All microglia are identified equally by both versions (green circles on the right), while the segments missed represent segments from background noise or unspecific expression of the fuorescent marker (red circles). Visualized using `v3d` [12].

to the ILP approach. The running time of the ILP solver increases much more rapidly, especially when the number of vertices exceeds 2000.

Scores of unconstrained versus constrained tree assignments. To quantify how the three-point condition affects the score of solutions (and thus the quality results), we compared the scores obtained using Algorithm 2 versus the scores obtained from the unconstrained tree assignment using the integer linear programming approach from [15], utilizing both synthetic data and real microscopic images of zebrafish brain from [15]. The results displayed in Fig. 3 indicate that while a small number of constrained scores achieve less than 30% of the unconstrained ones and can be considered to possibly loose critical segments, the majority of instances achieves a score of around 60% of the unconstrained score. Therefore, most of the results obtained using the constrained version are useful in practice. The score ratio is virtually constant across different signal-to-noise ratios in synthetic images with homogeneous background noise (data not shown).

6 Conclusion and Future Work

We have presented dynamic programming approaches to the tree assignment problem, which is of importance in bioimaging applications. From a theoretical point of view, constraining assignments to the three-point condition allows us to design a fast polynomial-time algorithm. We evaluated the practical implications of this, demonstrating that the dynamic programming approach enables us to solve large instances within a few seconds. Comparing the resulting scores with the unconstrained version suggests that solving the constrained version is sufficient in many cases. We plan to make the dynamic programming algorithm available in a future release of the `ct3d` software package for cell tracking and cosegmentation applications.

From an algorithmic point of view, investigating other variations of tree assignments is interesting from both theoretical and practical perspectives. We note that the more relaxed condition introduced by Lu et al. for the *less-constrained* edit distance [8] can also be applied to tree assignment. In that case, we can adopt the dynamic programming algorithm from Jiang et al. [5], for alignment of unordered rooted trees, to solve the less-constrained tree assignment problem. The drawback is that the algorithm has running time exponential in the degree of the trees. We are currently exploring the trade-off between the running time and the quality of the solution.

While beyond the scope of this more algorithmically focussed contribution, a more detailed evaluation of how restricting to the three-point condition affects results on both synthetic and real biological image data is desirable.

Acknowledgments. The work of HWL was supported in part by the National University of Singapore under ARF grant R-252-000-361-112.

References

1. Bille, P.: A survey on tree edit distance and related problems. Theoretical computer science 337(1-3), 217–239 (2005)
2. Canzar, S., Elbassioni, K., Klau, G.W., Mestre, J.: On tree-constrained matchings and generalizations. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6755, pp. 98–109. Springer, Heidelberg (2011)
3. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM) 34(3), 596–615 (1987)
4. Jiang, T., Lin, G., Ma, B., Zhang, K.: A general edit distance between RNA structures. Journal of computational biology 9(2), 371–388 (2002)
5. Jiang, T., Wang, L., Zhang, K.: Alignment of trees–an alternative to tree edit* 1. Theoretical Computer Science 143(1), 137–148 (1995)
6. Jones, R.: Component trees for image filtering and segmentation. In: IEEE Workshop on Nonlinear Signal and Image Processing (1997)
7. Leong, F.J., Brady, M., McGee, J.O.D.: Correction of uneven illumination (vignetting) in digital microscopy images. Journal of clinical pathology 56(8), 619 (2003)
8. Lu, C., Su, Z.Y., Tang, C.: A new measure of edit distance between labeled trees. In: Wang, J. (ed.) COCOON 2001. LNCS, vol. 2108, pp. 338–348. Springer, Heidelberg (2001)
9. Mattes, J., Richard, M., Demongeot, J.: Tree representation for image matching and object recognition. In: Bertrand, G., Couprie, M., Perroton, L. (eds.) DGCI 1999. LNCS, vol. 1568, pp. 298–309. Springer, Heidelberg (1999)
10. Mosig, A., Jäger, S., Wang, C., Nath, S., Ersoy, I., Palaniappan, K., Chen, S.S.: Tracking cells in Life Cell Imaging videos using topological alignments. Algorithms for Molecular Biology 4(1), 10 (2009)
11. Najman, L., Couprie, M.: Quasi-linear algorithm for the component tree. In: SPIE Vision Geometry XII, vol 5300, pp. 98–107 (2004)
12. Peng, H., Ruan, Z., Long, F., Simpson, J.H., Myers, E.W.: V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. Nature Biotechnology (2010)
13. Shapiro, B.A., Zhang, K.: Comparing multiple RNA secondary structures using tree comparisons. Computer applications in the biosciences: CABIOS 6(4), 309 (1990)
14. Sternberg, S.R.: Biomedical image processing. Computer 16(1), 22–34 (1983)
15. Xiao, H., Li, Y., Du, J., Mosig, A.: Ct3d: tracking microglia motility in 3D using a novel cosegmentation approach. Bioinformatics 27(4), 564 (2011)
16. Zhang, K.: A constrained edit distance between unordered labeled trees. Algorithmica 15(3), 205–222 (1996)

GASTS: Parsimony Scoring under Rearrangements

Andrew Wei Xu and Bernard M.E. Moret

Laboratory for Computational Biology and Bioinformatics, EPFL,
EPFL-IC-LCBB INJ230, Station 14, CH-1015 Lausanne, Switzerland
andywsureway@gmail.com, bernard.moret@epfl.ch

Abstract. The accumulation of whole-genome data has renewed interest in the study of genomic rearrangements. Comparative genomics, evolutionary biology, and cancer research all require models and algorithms to elucidate the mechanisms, history, and consequences of these rearrangements. However, rearrangements lead to NP-hard problems, so that current approaches, such as the MGR tool, are limited to small collections of genomes and low-resolution data of a few hundred syntenic blocks.

We describe the first algorithm for rearrangement analysis that scales up, in both time and accuracy, to modern high-resolution genomic data. Our main contribution is GASTS, an algorithm for scoring a fixed phylogenetic tree: given a tree and a collection of genomes, one for each leaf of the tree, each genome given by an ordered list of syntenic blocks, GASTS infers genomes for the internal nodes of the tree so as to minimize the sum, taken over all tree edges, of the pairwise genomic distances between tree nodes. We present the results of extensive testing on both simulated and real data showing that our algorithm runs several orders of magnitude faster than existing approaches and scales up linearly instead of exponentially with the size of the genomes involved; on the small instances that current approaches can complete in a day, our algorithm also returns much better scores. In simulations, our tree scores stay within 0.5% of the model value for trees up to 100 taxa and genomes of up to 10,000 syntenic blocks. GASTS enables us to attack heretofore unapproachable problems, such as accurate ancestral reconstruction of large genomes and phylogenetic inference for high-resolution vertebrate genomes, as we demonstrate on a set of vertebrate genomes with over 2,000 syntenic blocks.

1 Introduction

Genomic rearrangements were discovered early in the 20th century [20], but their systematic study started with the spread of sequencing technologies. In 1987 Day and Sankoff [7] proposed two major problems about rearrangements: the *edit distance*—given two genomes and a model of rearrangements, find the shortest sequence of rearrangements that transforms one input genome into the other; and the *median*—given three genomes, construct a fourth genome that minimizes the sum of its pairwise distances to the other three. The edit distance is computable in linear time for most models, while the median is NP-hard for most models [9]. Phylogenetic reconstruction from rearrangement data attracted attention, as rearrangements are "rare genomic events" [17] and thus might help resolve difficult questions about ancient branching patterns in evolution, but the computational complexity of parsimonious approaches precluded widespread application of the approach. The best available tool for the purpose, MGR [6] and its extensions, scales poorly in both accuracy and running time with genome size (to a few hundred syntenic blocks at most) and also with the expected length of the phylogenetic tree.

In this paper, we describe GASTS (Generalized Adequate Subtree Tree Scoring), a tree-scoring method based on generalized adequate subgraphs. Scoring a fixed tree given its leaf genomes is the core problem of phylogenetic inference, ancestral reconstruction, and all other uses of phylogenetic trees. The problem is NP-hard, as it subsumes the median problem. GASTS scales linearly with the expected length of the tree, and, in extensive simulation tests, returns tree scores within 0.5% of the model tree score. GASTS runs in seconds on datasets that MGR fails to complete in 24 hours and returns better scores on those datasets that MGR can complete. GASTS provides accurate values within the full range of practical applications in contemporary comparative genomics.

We test GASTS on real data and on simulations, the latter to assess scalability and absolute accuracy; in addition, we test its use within the contexts of both ancestral reconstruction and phylogenetic inference. Our simulations show that GASTS enables highly accurate tree reconstruction: even for difficult datasets, the expected error remains well below a single edge. On real data, GASTS enabled us to infer in just a few minutes phylogenies from high-resolution vertebrate data (over 2'000 syntenic blocks). Our new approach provides the kind of high-throughput tool needed today in comparative genomics and opens new areas of genomics to computational investigation.

2 Rearrangements and Phylogenetic Analysis

Rearrangement data was used in phylogenetic analysis 80 years ago by the Sturtevant and Dobzhansky [21]. Blanchette, Bourque, and Sankoff [5] introduced the first algorithmic approach to the reconstruction of a phylogenetic tree from rearrangement data, BPAnalysis. The algorithm seeks the tree and internal genomes which together minimize the total number of *breakpoints*—adjacencies present in one genome, but absent in the other. Moret *et al.* [14] reimplemented this approach in their GRAPPA tool and extended it to *inversion distances*—inversions are the best documented of the hypothesized mechanisms of genomic rearrangements; they also published the first studies of the median problem [13,18]. Their work focused on unichromosomal genomes; to handle multichromosomal genomes, Bourque and Pevzner [6] proposed MGR, based on GRAPPA's distance computations. Whereas BPAnalysis and GRAPPA search all trees and report the one with the best score, MGR uses a heuristic sequential addition method to grow the tree one species at a time.

Computing the parsimony score of a fixed tree for rearrangement data is NP-hard [9], even if the tree has only three leaves, and for any of breakpoint distance, inversion distance, and *DCJ distance*—a DCJ (Double-Cut-and-Join) operation changes two adjacencies at a time and provides a general framework that subsumes all other rearrangement operations [3]. The approach in BPAnalysis and GRAPPA is iterative refinement:

```
A. assign some arrangement to each internal node
B. repeat
     select an internal node x with a neighbor
       that was just assigned a new arrangement
     compute the median of the arrangements stored
       at the three neighbors of x
     if the median improves on the arrangement
       stored at x, assign the median to x
   until no change
```

The key problem here was long held to be the computation of medians, a problem that one of us started studying 8 years ago [13,18] and that we recently solved well enough for most practical purposes [15] in the context of unichromosomal genomes, using the concept of adequate subgraphs developed by one of us [26]. Using this median solver, we discovered that an equally crucial problem is the initialization phase: because local optima abound, an iterative refinement approach does well only when started with a very good initial assignment.

Our algorithmic contribution is a novel method for accurate initialization of internal (ancestral) genomes in a fixed tree. This contribution leads to an accurate tree-scoring algorithm, which in turn we use to run two different styles of phylogenetic inference, one through brute-force search (score all possible trees and retain the best) and one through incremental construction. One of us used GASTS to explore patterns beyond generalized adequate subgraphs on small phylogenetic trees [23], resulting in a collection of less constrained, yet more generally applicable configurations that can further reduce the running time of our approach on the hardest inputs.

3 A Heuristic for Multichromosomal Medians

Our approach is based on a fast and accurate heuristic for the inversion median that we recently developed [15] and now briefly review. This heuristic uses adequate subgraphs, developed by one of us [26,25], which provide optimality-preserving decompositions of the median problem and thus the basis for a divide-and-conquer algorithm for the median. Finding such decompositions in every instance remains an open problem; in practice, however, a few of the simplest decompositions suffice in almost every case—and when they do not, they can be supplemented by simple heuristics to produce highly accurate solutions [15].

We extend that work to handle the *capped* version of the underlying multiple breakpoint graphs—caps being necessary to move from unichromosomal to multichromosomal genomes [24]. At each step, our new algorithm either detects a capped adequate subgraph and decomposes the current instance into subproblems, or searches a polynomial number of ways to add one more adjacency into the median genome, selecting one choice according to a criterion that matches the definition of (capped) adequate subgraphs. Since this algorithm simply combines our design for inversion medians with Xu's extensions to capped multiple breakpoint graphs, we do not give a more detailed description, with one exception. Because DCJ-based approaches can produce extra circular chromosomes, our algorithm greedily merges such circular chromosomes with regular linear chromosomes so as to minimize the incremental increase of median scores. When tested on simulated data, these heuristics demonstrate very high accuracy, in line with our results for inversions [15].

4 Initialization with Adequate Subgraphs

In their paper on BPAnalysis [5], Sankoff and Blanchette proposed several initialization methods for step (A); both they and Moret *et al.* [14] settled on two approaches as the most promising. The *nearest-neighbor* approach assigns to each internal node the median of its three nearest leaves (one in each of the three adjacent subtrees, breaking ties within a subtree arbitrarily). The more complex *adjacency-parsimony* approach finds arrangements that minimize the number of adjacencies not already present in the data. This second method uses information in a more global manner than the first, although experiments by both sets of authors using breakpoint or inversion distances showed no gain in practice. Moreover, both initialization methods fail on difficult data, as the iterative refinement step (B) typically runs at most twice on each node—in other words, both initialization methods tend to start the algorithm in a local optimum, preventing any improvements and returning poor solutions. What is needed is an initialization that avoids local optima or uses only those very close to the global optimum—all of which argues for a better use of global information.

Our new initialization method put global information to use through adequate subgraphs and thus meshes well with the refinement phase of the scoring procedure, which uses adequate subgraphs to compute medians. We initialize internal nodes progressively: in order for an internal node to be a candidate for initialization, two of its three neighbors must be already initialized (or leaves). The third, while typically not be initialized, is not devoid of information, so our method summarizes the data available in the third subtree (rooted at the uninitialized neighbor) into a set of weighted adjacencies. Thus information used in initializing a node consists of two 0-1 sets of adjacencies from the two initialized neighbors and one weighted set of adjacencies from the third neighbor. A suitable choice of the node to be initialized is thus a generalized version of a median, one that takes into account the weighted nature of adjacencies in the third node.

4.1 Weighted Adjacencies and a Weighting Schema

We define a *perspective* at a node along one of its incident edges to be the subtree rooted at the other end of that edge. In Fig. 1, numbered nodes are uninitialized nodes and labelled nodes are leaves or initialized nodes. The subtree rooted at node 2 and extending rightward is the perspective at node 1 along edge z. We do not use the entire perspective in guiding the initialization of node 1, however; instead, we consider only the *directive nodes* in the perspective, that is, initialized nodes or leaves connected to the node of interest via a path of uninitialized nodes. In Fig. 1, nodes a, b, and c are initialized and connected to node 1 through paths of uninitialized nodes and so are directive nodes.

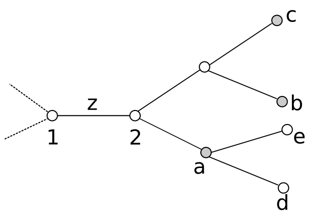


Fig. 1. A perspective at node 1 along edge z; shaded nodes are the directive nodes a, b, and c

Given a directive node g, we define the indicator function $I_x(g)$ to be 1 if adjacency x is present in g and 0 otherwise. We now define *weighted adjacencies* on perspective p at internal node i as follows: the weight w_x for each adjacency x is given by

$$w_x = \sum_{\text{directive nodes } g \text{ in } p} I_x(g) \cdot 2^{-d+1}$$

where d is the depth of node g—the number of edges on the path connecting nodes g and i. (The exponential decay reflects the exponential growth in the number of possible directive nodes in a perspective.)

Xu and Sankoff [26] showed that, under the DCJ model, if two of the three neighbors contain the same adjacency, then the median also contains it; under our weighting scheme, this property is preserved. Consider the situation depicted in Fig. 2. Say that nodes b, c, and e contain some adjacency x, while nodes a and d do not; should x be assigned to node 1? The presence of x in node b and the "fractional presence" of x in node 2 (the root of the perspective along edge z) together form a generalized adequate subgraph (for which see below) causing an assignment of x to node 1; in consequence, x also gets assigned to nodes 2 and 3.

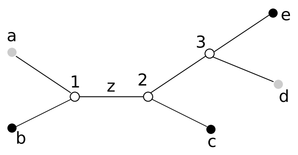


Fig. 2. A perspective at node 1 along edge z and nodes that share adjacency x (in black)

4.2 Using the Generalized Adequate Subgraphs

Adequate subgraphs capture optimal substructures, as defined and described in previous work by Xu *et al.* [15, 24, 26]. An adequate subgraph on n vertices is a connected subgraph of the multiple breakpoint graph, with which another set of edges can form at least $3n/4$ bicolored cycles. In the generalized version used in this paper, we count each cycle according to its weight, which we define to be the smallest weight along the cycle. Unlike the adequate subgraphs for the median problem, these *generalized adequate subgraphs (GAS)* do not have an optimality guarantee: they form the basis for heuristic assignment of the median in the initialization phase.

Using fractional rather than 0-1 weights greatly increases the number of GAS, so that detecting them can become a major computational task. However, each node to be initialized has two initialized (or leaf) nodes. The adjacencies in these two neighbors form bicolored cycles and paths in the breakpoint graph; we search for GAS only in these cycles and paths. This approach is not just efficient, but also reasonable: the genome to be assigned lies on the DCJ edit path between those two genomes and minimizes the average distance to the directive nodes. This genome may be slightly biased toward to the two known neighbors, when their edit distance is smaller than the real number of rearrangements. Since the genome we want to assign is well balanced between local and global information, the bias can be remedied in the second phase of the scoring procedure.

5 Testing Our New Tree-Scoring Method

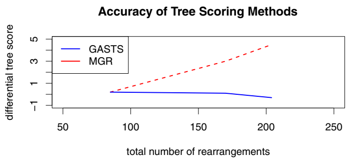
We compare GASTS with the only existing method that can handle multichromosomal data, MGR. However, MGR failed to complete within 24 hours of computation on almost every test case of medium to large size, so, in order to get some basic comparisons, we introduce a third method, purely a strawman to enable some comparison of GASTS scores and running times against at least one competitor on nontrivial datasets. This third method, denoted NNM, uses the standard nearest-neighbor initialization, but computes medians in both steps (A) and (B) using our GAS-based median solver.

We test tree-scoring quality and scalability on various model conditions with large genomes. Model conditions vary from 10 to 80 genomes, each made of up to 2,000 syntenic blocks. These models conditions are produced according to standard practice in phylogenetic reconstruction [10] as follows. We generate a rooted tree topology, assign a "genome" to the root, then simulate the evolution of the genome down the tree to the leaves. Trees are generated either following a standard birth-death process, in which case the length of each tree branch is determined during the construction of the tree topology, or by picking a tree uniformly at random among all distinct rooted trees on the assigned number of leaves, in which case we assign the same length to every edge of the tree. A tree with its branch lengths and root genome is a *model tree*; we view the length of branch of a model tree as the expected value of the length of that branch. From a model tree, we generate multiple datasets. To generate one dataset, we first assign real lengths to the edges of the tree by sampling from a Poisson distribution with a mean equal to the edge length in the model tree; we then "evolve" the root genome down the tree to obtain genomes at the leaves. To evolve a parent genome down to its child, we apply to the parent genome a number of rearrangements (chosen uniformly at random) equal to the length of that branch. The resulting leaf genomes, plus the tree topology (but not its branch lengths) form the dataset. Repeating the process on the same model tree produces new datasets. The tree with its edge lengths in each simulation is a *real tree*; the sum of its edge lengths is the *real tree score*. The sum of the edge lengths assigned to an instance by the tree-scoring procedure is the *inferred tree score*. The *differential tree score* is the difference between the inferred tree score and the real tree score. Since the rearrangement scenarios are random, the real tree score need not be the smallest score achievable for the tree.

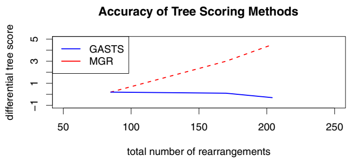
MGR's computational limitations led us to generate a first group of model conditions of modest scope: 10 genomes of size 100, with rearrangements limited to 80 to 340 inversions. As our own procedures, GASTS and NNM, can handle much larger trees and genomes, we generated a second group of model conditions with from 10 to 80 genomes, each with 20 linear chromosomes and a total of 2'000 syntenic blocks, using inversions, translocations, fissions, and fusions. In these datasets, model tree scores vary very widely, from 0.5 to 23 times the number of syntenic blocks. Finally, to test the computational scaling of GASTS, we generated a smaller number of large datasets, with 10 to 80 genomes, each made of 20 chromosomes with a total of 10'000 syntenic blocks.

On the first group of model conditions, GASTS and NNM ran in less than one second on every instance, whereas MGR took a very long time (often days) on over half of the instances—those derived from model trees with large scores. In order to run enough datasets, we set an arbitrary cutoff of 24 hours of computation per instance (on a dedicated CPU). Fig. 3 shows the difference between the inferred tree score and the real tree score for GASTS and MGR for the first group of model conditions. (Tree scores obtained from NNM were similar to, but less accurate than, those obtained from GASTS, so we do not present them here.) The horizontal axis denotes the real tree score, while the vertical axis denotes the differential tree score for each of GASTS and MGR. Inferred tree scores obtained by GASTS correctly trend downwards (for larger model tree scores we expect parsimony scores to be smaller than the model tree scores) and are consistently better than those obtained by MGR, which trend upwards, indicating increasing errors.

In the second group we used 50 different model conditions, with 10 datasets each. Here our comparison is between GASTS and NNM, because MGR could not complete any of these datasets within several weeks. On these datasets, GASTS runs in a few seconds for the smaller datasets and in a few minutes for the larger ones. Since both methods use the same median solver and share the same refinement step, the results indicate the differences in the initialization method. The NNM approach suffers when the number of leaves grows, as it must then compute medians of very divergent leaves, which can take significant time; in contrast, GASTS initialization ensures that every median computation is a median of three neighboring genomes. The speed difference is large enough that, with the NNM approach, we could not complete instances for trees with 20 or more taxa. Fig. 4 shows the results, including higher-resolution plots of the final differential tree scores for GASTS. For both methods, we report the differential



(a) on birth-death trees



(b) on random trees

Fig. 3. Difference between inferred and model tree scores for GASTS and MGR, only on those datasets completed by MGR within 24 hours

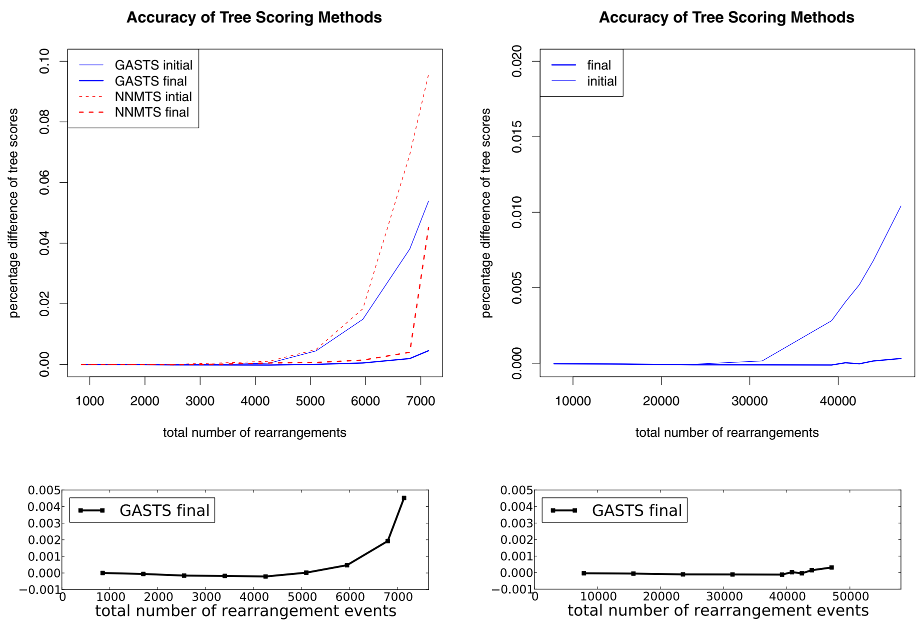
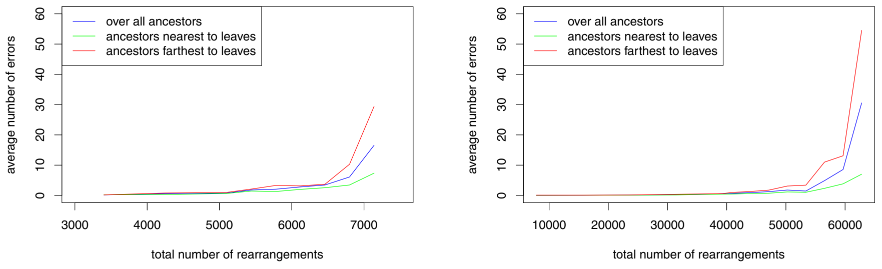


Fig. 4. Accuracy of GASTS and NNM, for 10-taxon trees (left) and 80-taxon trees (right), as a function of the total number of rearrangement events. Thin lines show initial scores, thick lines final scores. The lower plots zoom in on the final score produced by GASTS.

tree score right after initialization and at the completion of the scoring procedure. Our initialization method clearly dominates the NNM method, particularly for large model tree scores. The second phase of scoring—the iterative refinement—has little effect for smaller distances, but quite dramatic ones for large distances, showing that our initialization method helps the refinement procedure in avoiding local optima. Note that the inferred score keeps tracking the real score even at very large evolutionary distances, as evidenced in the lower plots. Thus GASTS is both highly accurate and very robust—in particular, tree scores inferred by GASTS can meaningfully be compared.

6 Ancestral Genome Reconstruction

In scoring trees, we assign arrangements to the internal nodes. Such assignments should not be confused with true ancestral reconstruction, as they do not obey specific biological constraints, but simply optimize a distance function. Yet many ancestral reconstruction approaches to date use median computations as part of their guidance. Given the high accuracy of our scoring algorithm, we may expect it to assign arrangements to internal nodes that come quite close to the "true" ancestral genomes. Since our simulations create these "true" ancestral genomes, we can compare them to those reconstructed by our scoring procedure. Indeed, the genomes assigned by GASTS are very close to the true ancestral genomes, as shown in Fig. 5. Interestingly, ancestral nodes farthest from the leaves are not much worse than those closest to the leaves, which also



(a) on random tree with 10 species (b) on random tree with 80 species

Fig. 5. Average total DCJ distance between genomes assigned to internal nodes and the corresponding "true" ancestors

supports the quality of the scoring. (For the largest distances, direct reconstruction fails due to the enormous number of optimal solutions, as was shown for a collection of gamma proteobacteria [8].) Over all 22 model conditions, the average total distance was less than 30; in 20 of the 22 model conditions it was less than 10 and, in 10 model conditions, it was less than 1. Thus the high accuracy of GASTS indeed provides a good starting point for ancestral reconstruction.

7 Phylogenetic Reconstruction

With good tree scoring, several reconstruction methods used with sequence data can be adapted to rearrangement data. We present experimental results for two different uses of GASTS in phylogenetic inference: an exhaustive search method that scores every tree and returns the best, and a heuristic method that builds a tree by sequential addition. Exhaustive tree search may give the best results, but is forcibly limited to at most 20 taxa—we did not attempt to use the many speed-up mechanisms of GRAPPA, as our aim was to test GASTS, not to produce a better reconstruction algorithm. Adding one taxon at a time is a very simple heuristic, but one with a long history in phylogenetic analysis, including incremental parsimony [4] and quartet-puzzling [19] for sequence-based data; in rearrangement-based phylogenetic work, MGR uses this approach. Unlike these three methods, our sequential addition method rescores every tree after each addition step. We compare these two approaches with MGR rather than with GRAPPA, so as to test performance on multichromosomal as well as unichromosomal data.

We used datasets of modest size so as to allow a comparison with MGR. Rearrangement include: (i) inversions on unichromosomal circular genomes; (ii) inversions, translocations, fusions, and fissions on multichromosomal genomes; and (iii) 80% inversions and 20% transpositions on unichromosomal circular genomes. We tested trees generated from three different models: the birth-death model, the random model, and the beta-splitting model [2] with $\beta = -1$. We report results only for the birth-death and uniform random model, as results for the beta-splitting model were similar to those for the uniform random model. For each dataset completed by MGR in 24 hours, we compute the *Robinson-Foulds* (RF) error rate to the real tree—half of the number of edges present in one tree, but not in the other [16], divided by the number of internal edges in the model tree.

MGR finished only the smaller datasets in each group, taking about 20 minutes per dataset with tree scores of 85 and over 12 hours per dataset with tree scores of 200, its running time increased exponentially as a function of the tree score. Our exhaustive search method took just one minute on each dataset, except for a few datasets with tree scores of 340, where it took around 12 minutes. Finally, our sequential addition method took less than one second per dataset, except for a few datasets with tree scores of 340, where it took 4 seconds. Table 1 shows the error rates for the three methods averaged over 6 model conditions, using birth-death trees. All three show reasonable accuracy, with the exhaustive search the most accurate, the sequential addition second, and MGR third. The main drawback of MGR here is running time (empty table entries correspond to model conditions under which MGR could not complete more than a few test instances), not accuracy: the phylogenetic signal in the data is strong enough that correct inferences can be made even if the reconstruction method does not take full advantage of the data. The introduction of transpositions worsened the results for all three approaches, most significantly for MGR—as might be expected, since a transposition takes two operations in the DCJ framework used in our methods, but three operations in the inversion framework used by MGR.

Table 1. RF error rates (in %)on datasets of 10 genomes of size 100 on birth-death trees, as a function of the number of rearrangements

(a) circular genome, inversions & transpositions

	85	170	204	238	255	340
exhaustive	0.0	0.0	0.0	5.2	2.6	5.2
sequential	0.0	0.0	1.8	5.2	4.3	14.3
MGR	0.0	7.9	7.1			

(b) linear genome, all rearrangements

	85	170	204	238	255	340
exhaustive	0.0	0.0	0.0	0.0	2.9	5.7
sequential	0.0	1.4	0.0	0.0	4.3	2.9
MGR	0.0					

To test the accuracy of the sequential addition heuristic on large trees with large genomes, we generated eight model conditions (again with 10 datasets each) using birth-death trees on 80 taxa, with genomes of 2,000 syntenic blocks allotted among 20 linear chromosomes. Four of the model conditions use a mix of 20% transpositions,

Table 2. Average RF error rates (in %) and running times (in mins.) for sequential addition on datasets of 80 genomes of 20 chromosomes with 2,000 blocks as a function of the number of rearrangements

(a) no transpositions

	5,000	10,000	15,000	20,000
error	0.0	0.0	0.0	0.78
time	21.0	21.8	26.7	88.3

(b) 20% transpositions

	5,000	10,000	15,000	20,000
error	0.0	0.0	1.56	1.95
time	22.9	28.7	74.3	845.0

while the other four do not use transpositions; the total number of rearrangement events ranges ranges from 5,000 to 20,000. Table 2 shows error rates and running times; the approach shows excellent scalability and good accuracy, with error rates consistently smaller than 2%.

8 Applications on Real Data

We applied our exhaustive approach to two biological datasets. The first is quite small: the well studied Campanulaceae chloroplast dataset, with 13 taxa, each genome a circular chromosome with 105 genes. The second is a collection of 8 vertebrate genomes (7 mammals and chicken), from a 13-way genomic alignment downloaded from ENSEMBL, from which we retained the best assembled genomes. We then generated synteny blocks at five different resolutions: 1Kbp, 3Kbp, 10Kbp, 30Kbp, and 100Kbp—meaning that the resolution value was used as a lower bound on the size of acceptable blocks. (We ignored contigs that contained a single synteny block.)

Previous studies on the Campanulaceae dataset reported best tree scores of 64 inversions and 64 DCJ operations [1,11,13]. Our approach improved the DCJ tree score by finding 138 trees with a score of 63, all with a different topology from the tree with DCJ score of 64 reported in [1]. The vertebrate set has not been analyzed by anyone as whole genomes at these resolutions, so direct comparisons are not possible. Nor is comparison with sequence-based analyses fruitful at this stage, since the use of rearrangement data in phylogenetic analysis is too immature for an interpretation of the resulting trees; moreover, the decomposition of whole genomes into syntenic blocks is itself a poorly resolved problem, one aggravated by the incomplete assembly of many of these genomes. Instead, our purpose with these datasets is to demonstrate the scalability of our approach and thus pave the way for detailed studies. Our exhaustive approach completed each of the five datasets in a few minutes and returned the same tree for each (the generally accepted tree), illustrated in Figure 6.

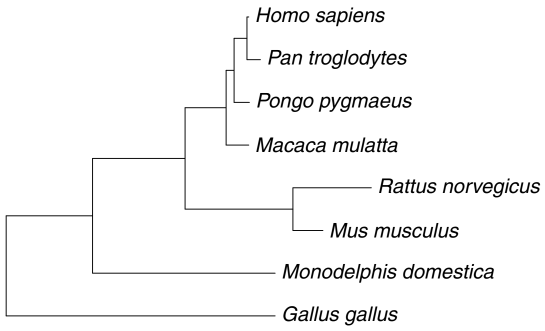


Fig. 6. Phylogeny of 8 vertebrates

9 Conclusion

We presented a new approach to phylogenetic analysis of rearrangement data that integrates past and recent work through various adaptations (such as generalized adequate subgraphs) and includes GASTS, a fast, highly accurate, and surprisingly robust scoring method for a fixed tree. We presented experimental results demonstrating that our scoring procedure scales gracefully to trees far larger than anything used to date with gene-order data, as well as to evolutionary distances that are well into saturation, all while returning tree lengths in a very narrow interval around the true tree length (generally within less than 0.1% and even in the worst cases within 0.5%). We tested our reconstruction methods under simulation and on datasets of whole genomes; the simulations indicate that the reconstruction is perfect in almost all cases, while the whole-genome datasets demonstrate the power of the method on a dataset of vertebrates with up to 10,000 markers, a scale that had been entirely out of reach until now. While our approach does not yet handle duplications and losses and so must be used with either simple genomes like organelles or large genomes represented by sequences of unique syntenic blocks (as is often done for vertebrates), it makes it possible, for the first time, to conduct nontrivial phylogenetic analyses from high-resolution genomic data.

References

1. Adam, Z., Sankoff, D.: The ABCs of MGR with DCJ. Evol. Bioinf. Online 4, 69–74 (2008)
2. Aldous, D.J.: Stochastic models and descriptive statistics for phylogenetic trees, from Yule to today. Stat. Sci. 16, 23–34 (2001)
3. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 163–173. Springer, Heidelberg (2006)
4. Bininda-Emonds, O.R.P., Brady, S.G., Kim, J., Sanderson, M.J.: Scaling of accuracy in extremely large phylogenetic trees. In: Proc. 6th Pacific Symp. on Biocomputing (PSB 2001), pp. 547–558. World Scientific Pub., Singapore (2001)
5. Blanchette, M., Bourque, G., Sankoff, D.: Breakpoint phylogenies. In: Miyano, S., Takagi, T. (eds.) Genome Informatics, pp. 25–34. Univ. Academy Press, Tokyo (1997)
6. Bourque, G., Pevzner, P.: Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Res. 12, 26–36 (2002)
7. Day, W.H.E., Sankoff, D.: The computational complexity of inferring phylogenies from chromosome inversion data. J. Theor. Biol. 127, 213–218 (1987)
8. Earnest-DeYoung, J., Lerat, E., Moret, B.M.E.: Reversing gene erosion: reconstructing ancestral bacterial genomes from gene-content and gene-order data. In: Jonassen, I., Kim, J. (eds.) WABI 2004. LNCS (LNBI), vol. 3240, pp. 1–13. Springer, Heidelberg (2004)
9. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. MIT Press, Cambridge (2009)
10. Hillis, D.M.: Approaches for assessing phylogenetic accuracy. Syst. Biol. 44, 3–16 (1995)
11. Larget, B., Kadane, J.B., Simon, D.L.: A Markov chain Monte Carlo approach to reconstructing ancestral genome arrangements. Mol. Biol. Evol. 22, 486–489 (2002)
12. Miklós, I., Mélykúti, B., Swenson, K.M.: The metropolized partial importance sampling MCMC mixes slowly on minimal reversal rearrangement paths. ACM/IEEE Trans. on Comput. Bio. & Bioinf. 7(4), 763–767 (2010)

13. Moret, B.M.E., Siepel, A.C., Tang, J., Liu, T.: Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 521–536. Springer, Heidelberg (2002)
14. Moret, B.M.E., Wyman, S.K., Bader, D.A., Warnow, T., Yan, M.: A new implementation and detailed study of breakpoint analysis. In: Proc. 6th Pacific Symp. on Biocomputing (PSB 2001), pp. 583–594. World Scientific Pub., Singapore (2001)
15. Rajan, V., Xu, A.W., Lin, Y., Swenson, K.M., Moret, B.M.E.: Heuristics for the inversion median problem. In: Proc. 8th Asia Pacific Bioinf. Conf. (APBC 2010). BMC Bioinformatics, vol. 11(suppl. 1), p. S30 (2010)
16. Robinson, D.R., Foulds, L.R.: Comparison of phylogenetic trees. Math. Biosci. 53, 131–147 (1981)
17. Rokas, A., Holland, P.W.H.: Rare genomic changes as a tool for phylogenetics. Trends in Ecol. and Evol. 15, 454–459 (2000)
18. Siepel, A.C., Moret, B.M.E.: Finding an optimal inversion median: Experimental results. In: Gascuel, O., Moret, B.M.E. (eds.) WABI 2001. LNCS, vol. 2149, pp. 189–203. Springer, Heidelberg (2001)
19. Strimmer, K., von Haeseler, A.: Quartet puzzling: A quartet maximum likelihood method for reconstructing tree topologies. Mol. Biol. Evol. 13, 964–969 (1996)
20. Sturtevant, A.H.: A crossover reducer in Drosophila melanogaster due to inversion of a section of the third chromosome. Biol. Zent. Bl. 46, 697–702 (1926)
21. Sturtevant, A.H., Dobzhansky, T.: Inversions in the third chromosome of wild races of D. pseudoobscura and their use in the study of the history of the species. Proc. Nat'l Acad. Sci., USA 22, 448–450 (1936)
22. Tang, J., Moret, B.M.E.: Scaling up accurate phylogenetic reconstruction from gene-order data. In: Proc. 11th Int'l Conf. on Intelligent Systems for Mol. Biol (ISMB 2003). Bioinformatics, vol. 19, pp. i305–i312 (2003)
23. Xu, A.W.: On exploring genome rearrangement phylogenetic patterns. In: Tannier, E. (ed.) RECOMB-CG 2010. LNCS, vol. 6398, pp. 121–136. Springer, Heidelberg (2010)
24. Xu, A.W.: DCJ median problems on linear multichromosomal genomes: Graph representation and fast exact solutions. In: Ciccarelli, F.D., Miklós, I. (eds.) RECOMB-CG 2009. LNCS, vol. 5817, pp. 70–83. Springer, Heidelberg (2009)
25. Xu, A.W.: A fast and exact algorithm for the median of three problem—a graph decomposition approach. J. Comput. Biol. 16(10), 1369–1381 (2009)
26. Xu, A.W., Sankoff, D.: Decompositions of multiple breakpoint graphs and rapid exact solutions to the median problem. In: Crandall, K.A., Lagergren, J. (eds.) WABI 2008. LNCS (LNBI), vol. 5251, pp. 25–37. Springer, Heidelberg (2008)

OMG! Orthologs in Multiple Genomes – Competing Graph-Theoretical Formulations

Chunfang Zheng[1,2], Krister Swenson[1,2], Eric Lyons[3], and David Sankoff[1]

[1] Department of Mathematics and Statistics, University of Ottawa
[2] Département d'informatique et de recherche opérationnelle, Université de Montréal
[3] iPlant, Department of Plant Sciences, University of Arizona

Abstract. From the set of all pairwise homologies, weighted by sequence similarities, among a set of genomes, we seek disjoint orthology sets of genes, in which each element is *orthogonal* to all other genes (on a different genome) in the same set. In a graph-theoretical formulation, where genes are vertices and weighted edges represent homologies, we suggest three criteria, with three different biological motivations, for evaluating the partition of genes produced by deletion of a subset of edges: i) minimum weight edge removal, ii) minimum degree-zero vertex creation, and iii) maximum number of edges in the transitive closure of the graph after edge deletion. For each of the problems, all either proved or conjectured to be NP-hard, we suggest approximate and heuristic algorithms of finding orthology sets satisfying the criteria, and show how to incorporate genomes that have a whole genome duplication event in their immediate lineage. We apply this to ten flowering plant genomes, involving 160,000 different genes in given pairwise homologies. We evaluate the results in a number of ways and recommend criterion iii) as best suited to applications to multiple gene order alignment.

1 Introduction

Multiple alignment of the gene orders in sequenced genomes is an important and timely problem in comparative genomics [1,2,3,4]. A key aspect is the construction of disjoint orthology sets of genes, in which each element is orthologous to all other genes (on different genomes) in the same set. Approaches differ as to the nature and timing and relative importance of sequence alignment, synteny block construction, and paralogy resolution in constructing these sets. We argue that these considerations are best integrated in the construction of *pairwise* synteny blocks as a first step, followed by the conflation of the pairwise orthologies into larger sets. The advantages of this are the availability of finely tuned pairwise synteny block software (e.g., SynMap in the CoGe platform [5,6]), the possibility of dealing with paralogs dating from ancient whole genome duplication (WGD) in a natural way, and the opportunity to dispense with thresholds or other arbitrary settings during the construction of the orthology sets themselves. The task of discerning the orthology sets becomes a purely algorithmic problem on graphs.

Here we distinguish three variants of this Orthologs in Multiple Genomes (omg) problem, differentiated by their objective functions and their biological justifications. All are expected to be hard, from the algorithmic point of view, although some have available approximation algorithms. In Section 2, we first define the different formulations of the omg problem and discuss their biological interpretations. We then present and analyze the algorithms we design for each formulation, including variants of the omg problem incorporating paralogy data from genomes known to be descendants of wgd events.

Our approach, though widely applicable, was developed within the context of flowering plant genomics, an evolutionary domain characterized by recurrent wgd events. The massive data set we analyze is described in Section 3.

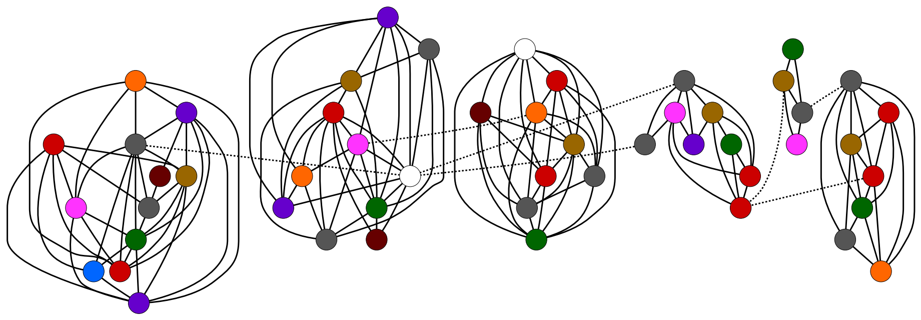


Fig. 1. Homology set, showing "erroneous" edges between orthology sets

In Section 4, we compare the results of applying the algorithms to a large data set of homologies among some 160,000 plant genes drawn from ten eudicotyledon genomes. We compare the results of the three methods when evaluated by the other two objectives, and by assessing the compatibility of the orthology sets with the phylogenetic tree that is assumed to have generated them. We find that some of the orthology sets recovered by each method are also recovered by the other two, but each method also finds many sets specific to that method.

Section 5 contains the conclusions, including reflections on restrictions on accessibility of genome data for the purposes of comparative genomics. We conclude that one of the three approaches produces somewhat bigger orthology sets, so that insofar as these sets are biologically validated, this method can be recommended for purposes of multiple gene order alignment.

2 The Competing Formulations and Their Algorithms

The pairwise homologies SynMap provides for all pairs of genomes constitute the set of edges E of the *homology graph* $H = (V, E)$, where V represents the set of genes in any of the genomes participating in at least one homology relation. In addition there is a weight $w(e) \in (0, 1]$ associated with each edge $e \in E$, representing a protein similarity score. Let $H = H_1 \cup \cdots \cup H_s$ represent the decomposition of H into connected components. Since we expect SynMap to resolve all or most paralogies, ideally all the genes in each H_i should be orthologous. There should be *at most* one gene from each genome in such an orthology set, or at most two duplicate genes for genomes that descend from a WGD event. In practice, however, as in Fig. 1, there may be several genes from the same genome in an H_i, apparent paralogies, which we shall consider erroneous due to spurious homologies (edges) in the input. The problem we address, then, is how to convert H into a new graph $O = O_1 \cup \cdots O_t$ with the *orthogonality* property desired of each connected component O_j, namely that it contain no paralogs, except for duplicates in genomes descended from WGDs.

Definition 1. *A graph $O = (V, E)$ with vertices in c colour classes is an orthogonal partition if each of its components contains at most one vertex of any one colour.*

Given any graph $H = (V, E)$ with coloured vertices, it can be converted into an orthogonal partition $O = (V, E \setminus E')$ by deleting a subset E' of edges from E.

Problem 1. Given a criterion $\kappa = \kappa(V, E, E')$, where $(V, E \setminus E')$ is an orthogonal partition, the OMG problem is to find a subset $E' \subset E$ that optimizes κ.

Definition 2. *Let $c = c_1 + c_2$ where there are c_2 distinguished colours called* WGD *colours. The graph $O = (V, E)$ is a* WGD*-orthogonal partition if each of its components contains at most two vertices of any of the c_2 distinguished colours, and at most one vertex of any of the c_1 remaining colours.*

Problem 2. Given $c_2 \leq c$ distinguished colour classes and criterion $\kappa = \kappa(V, E, E')$, where $(V, E \setminus E')$ is an WGD-orthogonal partition, the OMG problem is to find a subset $E \subset E'$ that optimizes κ.

In Sections 2.1,2.2 and 2.3 we will discuss motivations for three different criteria κ and present algorithms for each one, for both the usual definition of orthogonality (Definition 1) and for the WGD version (Definition 2).

2.1 Minimum Weight Orthogonal Partition (MWOP)

Our first approach is simply to delete a set of edges E' of minimum weight. This definition of κ is motivated by the desire to conserve as many of the homology inferences in the input data as possible, and to discard as noise as few as possible. This is a NP-hard graph problem, Minimum Weight Orthogonal Partition (MWOP), for which He *et al.* have given an approximation algorithm [7].

The algorithm iteratively merges vertices into orthogonal sets using the maximum weight bipartite matching at each step; the result of a matching between two colours produces orthogonal sets which each can be considered as single vertices with two colours. Now a second matching can be found, and so on. We used an auction routine for maximum weight bipartite matching [11,12].

The output of the algorithm depends on the order in which the vertices are merged. In our version, this order is determined by phylogeny. Thus, part of the input is a rooted, binary tree T, with each leaf corresponding to one of the given colours . The biological motivation is that a homology relation between genes in closely related genomes is less likely to be spurious than in distant relatives.

The case of no WGD descendants (Problem 1).

```
1. for each genome $i$ with $n_i$ genes, $n_m = \max_i n_i$, define sets
       $\mathbf{V^i} = \{V_1^{\{i\}} = \{v_1\}, \cdots, V_{n_i}^{\{i\}} = \{v_{n_i}\}, V_{n_i+1}^{\{i\}} = \emptyset, \cdots, V_{n_m}^{\{i\}} = \emptyset\}.$
2. All the leaves of $T$ are eligible to merge. All ancestral nodes are ineligible.
3. while there remain unmerged but eligible sister nodes (same immediate
   parent) in the tree $T$.
   (a) choose eligible sister nodes $i$ and $j$.
   (b) construct a complete bipartite graph between $\mathbf{V^i}$ and $\mathbf{V^j}$.
   (c) if there is an edge $uv$, with $u \in V_a^{\{i\}}, v \in V_b^{\{j\}}$ set
           $w(V_a^{\{i\}}, V_b^{\{j\}}) = \sum_{u \in V_a^{\{i\}}, v \in V_b^{\{j\}}} w(uv),$
       otherwise $w(V_a^{\{i\}}, V_b^{\{j\}}) = 0$.
   (d) find the maximum weight matching for the bipartite graph.
   (e) for each pair $V_a^{\{i\}} V_b^{\{j\}}$ in the matching, set
           $V_a^{\{i\}} \leftarrow V_a^{\{i\}} \cup V_b^{\{j\}}.$
   (f) the new $\mathbf{V^i}$ is associated with the ancestral node of $T$ that is the
       immediate parent of $i$ and $j$, which now becomes eligible to merge;
       $V_b^{\{j\}}$ is now disregarded.
4. the remaining set $V^i$ corresponds to $O$ (i.e. $O_1 = V_1^{\{i\}}$, $O_2 = V_2^{\{i\}}$, etc.).
```

WGD descendants allowed. (Problem 2.) If genome j is a WGD descendant, then after Step 3 (e)

```
i.   if a vertex $V_a^{\{i\}}$ is matched to a vertex $V_b^{\{j\}}$ in genome $j$, where
     $w(V_a^{\{i\}} V_b^{\{j\}}) > 0$, then set $V_b^{\{j\}} \leftarrow \emptyset$.
ii.  construct a complete bipartite graph for $\mathbf{V^i}$ and the modified genome $j$.
iii. find the maximum weight matching for this graph.
iv.  for any two positively weighted matching edges $e_1 = V_x^{\{i\}} V_y^{\{j\}}, e_2 =$
     $V_u^{\{i\}} V_v^{\{j\}}$ that share a gene $z$, remove the edge with less weight.
v.   for all remaining edges $e$ merge the vertex in genome $j$ with the one in $\mathbf{V^i}$
     if a positively matched edge ($w > 0$) exists between these two vertices, and
     remove that $j$ genome gene from any other $\mathbf{V^i}$ vertex it may already be in
     (from the merge in step 3 (e)).
```

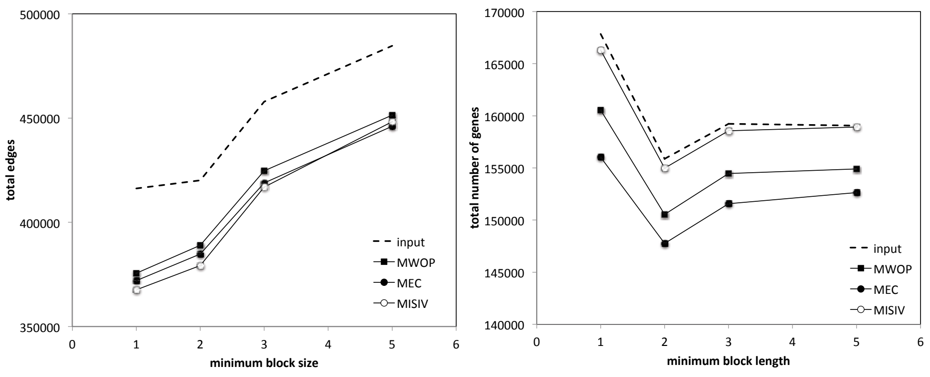


Fig. 2. Edges retained and degree-zero vertices created by the three OMG methods

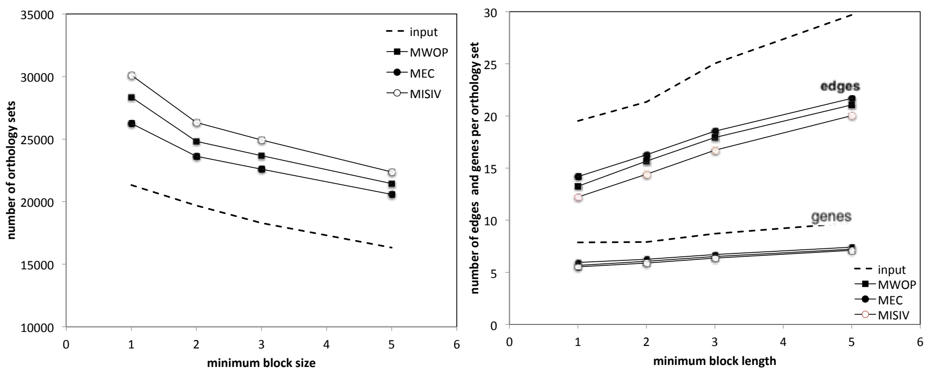


Fig. 3. Number and size of orthology sets produced by the three OMG methods

2.2 Minimize Singleton Vertices (MISIV)

Deleting edges in E' can create *singletons*, degree zero vertices. Each of these trivial components contains no orthology information, and is of little use in comparative genomic applications such as multiple gene order alignment. Our second criterion, then, is to minimize the number of vertices of degree zero created by edge removal. This definition of κ is designed to keep homology information on as many genes as possible in the data set. We seek an orthogonal partition $O = (V_1, E_1) \cup \cdots \cup (V_t, E_t)$ to minimize $|\{E_i | E_i = \emptyset\}|$. We are not aware of previous algorithms for this problem, and conjecture that it is NP-hard.

The case of no WGD descendants (Problem 1). We first initialize $E' = \emptyset$, subsets $V_v = \{v\}$ for each $v \in V$, and the relation c such that $V_v \mathrel{\mathsf{c}} V_u$ if u and v have the same colour and $V_v \not\mathrel{\mathsf{c}} V_u$ if not. We set $w(uv) = 0$ if $uv \notin E$. Without ambiguity, we extend w to be a weight on pairs of sets, and $w(\{u\}, \{v\}) = w(uv)$.

```
1. while there remains a subset $V_v$ consisting of a single vertex of degree 1,
   and an edge $uv \in E \setminus E'$ for some vertex $u \in V_x$, where $V_v \not\mathsf{c}\, V_x$, and $w(uv)$
   is maximum over all $V_v$, do the following:
   (a) $V_x \leftarrow V_x \cup V_v$.
   (b) if $V_x \,\mathsf{c}\, V_z$ or $V_v \,\mathsf{c}\, V_z$ for any set $V_z$, then we set $V_x \,\mathsf{c}\, V_z$.
   (c) for all $V_w$ consisting of a single vertex of degree 1, where $zw \in E \setminus E'$,
       $z \in V_x$, $V_w \,\mathsf{c}\, V_x$, delete edge $zw$, i.e. $E' \leftarrow E' \cup \{zw\}$
   (d) $V_v \leftarrow \emptyset$.
2. while there remains a subset $V_v$ consisting of a single vertex $v$, and an
   edge $uv \in E \setminus E'$ for some vertex $u \in V_x$, where $V_v \not\mathsf{c}\, V_x$,
   (a) Construct the subgraph of $(V, E \setminus E')$ induced by all $v$ satisfying these
       conditions. Find the maximum weight matching of these subsets.
   (b) for each pair $V_x$ and $V_y$ in the matching,
       merge$[V_x, V_y]$ :
         i. $V_x \leftarrow V_x \cup V_y$.
        ii. if $V_x \,\mathsf{c}\, V_z$ or $V_y \,\mathsf{c}\, V_z$ for any set $V_z$, then we set $V_x \,\mathsf{c}\, V_z$ and
            $w(V_x, V_z) = 0$. If there is an edge $e$ joining any vertex in $V_x$ and
            any vertex in $V_z$, delete it, i.e., $E' \leftarrow E' \cup \{e\}$.
       iii. for all $V_z$, $V_z \not\mathsf{c}\, V_x$, set $w(V_x, V_z) = \sum_{\{uv|u \in V_x, v \in V_z\}} w(u, v)$,
               if $w(V_x, V_z) > 0, E = E \cup \{V_x V_z\}$.
        iv. $V_y \leftarrow \emptyset$.
3. while there remain at least two subsets $V_x \not\mathsf{c}\, V_y$, and vertices $u \in V_x, v \in V_y$,
   where $uv \in E \setminus E'$,
   (a) Find the maximum weight matching among all these subsets.
   (b) for each pair $V_x$ and $V_y$ in the matching,
       merge$[V_x, V_y]$ (Steps 2 (b) i.–iv. above)
4. relabel the remaining sets $O_1, \cdots, O_t$; these contain the vertices of the
   required components of $O$.
```

The strategy of this heuristic is to irreversibly enlarge the components by first adding the vertices most vulnerable to becoming degree zero through edge deletion, namely those of degree one. After as many of these as possible (or a combination of them having greatest weight) are thus "protected", we then try to protect as many others as possible through a series of maximum weight matches in the subgraph induced by unprotected vertices. Step 3 may merge some sets of vertices without any effect on the objective function (minimum number of degree zero vertices), but in a way that tends to improve our result with respect to other objective functions (fewer and larger components; fewer, or lesser combined weight of, edges deleted).

WGD descendants allowed. (Problem 2.) The algorithm is easily extended to genomes that have paralogs resulting from WGD. The color relation c originally served to block any merger of two sets that would result in two paralogs in the same set. It needs only to be reinterpreted to block mergers resulting in three paralogs in the same set from a WGD descendant.

2.3 Maximum Edges in Transitive Closure (MEC)

The understanding of orthologous genes in two genomes as originating in a single gene in the most recent common ancestor of the two species leads logically to transitivity as a necessary property of the orthology relation. If gene x in genome X is orthologous both to gene y in genome Y and gene z in genome Z, then y and z must also be orthologous, even if SynMap does not detect this homology.

This motivates our third criterion for $O = O_1 \cup \cdots O_t$, namely that the weights of the edges in the transitive closure of O (or in all the cliques generated by the components O_i) be maximized. In other words, this definition of criterion κ maximizes the the sum of the weights over $\sum_1^t \binom{|O_i|}{2}$ edges, preferring to create a few large orthology sets rather than many smaller ones with the same total number of edges. Again, we are not aware of any previous algorithm for this problem, but conjecture it to be NP-hard.

Let $\bar{H} = (V, \bar{E})$ be the transitive closure of graph H. To obtain $\bar{H}$ we raise its adjacency matrix M_H (including 1's on the diagonal) to successively higher powers until convergence to some M_H^r. This could be accelerated using Warshall's algorithm [8]. (N.B. $r \leq$ diameter(H).) Without loss of generality we may assume H is connected, so $\bar{H}$ is a complete graph and all elements of M_H^r are non-zero. Elements of this matrix thus obtained should represent indirectly inferred orthologies as discussed above, but there may in fact be many paralogies. To remedy, this we first examine the star subgraph $s(v)$ of $\bar{H}$ containing $\nu(v)$ vertices, namely v, its $\nu(v) - 1$ neighbours, and the $\nu(v) - 1$ edges connecting the former to the latter.

Let $c(v) \geq 1$ be the number of distinct colours among the vertices in $s(v)$. Let $F(E) = \sum_{v \in V} c(v)$.

The case of no WGD descendants (Problem 1).

```
1. set $E' = \emptyset$.
2. while there are still some $v \in V$ where $\nu(v) > c(v)$,
   (a) find the edge $e \in E \setminus E'$ that maximizes
         $$F(E \setminus E'') = \sum_{v \in V} c(v), \text{where } E'' = E' \cup \{e\}$$
   (b) if there are several such $e$, find the one that minimizes
         $$F^+(E \setminus E'') = \sum_{(V, E \setminus E'')} \nu(v) - c(v).$$
   (c) if there are still several such $e$, find one with minimum $w(e)$.
   (d) $E' \leftarrow E' \cup \{e\}$
3. relabel as $O_1, \cdots, O_t$ the disjoint components created by deleting edges.
   These contain the vertices of the required components of $O$.
```

Implicit in each greedy step is an attempt to create large orthology sets. If the deleted edges create two partitioned components, i.e., each with no internal paralogy, then the increment in F will be proportional to the sum of the squares of the number of vertices in each one. This favours a decomposition into one large and one small component rather than two equal sized components.

WGD descendants allowed. (Problem 2.) To handle paralogs of WGD origin, the definition of $c(v)$ must be amended to take account an allowance of 2 vertices of the same colour in $s(v)$ if these are from the appropriate genomes. And the condition in Step 2 must require that at most two vertices be contained in $s(v)$ of any one colour, and only if these involve WGD descendants.

2.4 Issues of Accuracy, Deliberate Bias and Interpretation

In the next section, we focus more on the systematically divergent output generated by the different objective functions κ than on accuracy issues of the algorithms themselves. We know that the MWOP algorithm [7] is an approximation algorithm with a large approximation constant. Notwithstanding this uncertainty, it works well on small examples, as confirmed by our own testing. The other heuristics also satisfy the objective functions, or come very close, in small scale tests. In addition, they are efficient, an important consideration when there are many components H_i each containing hundreds of edges. Most importantly, however, the three algorithms will be seen to produce different inventories of orthology sets, each with a bias for meeting a particular biological motivation; MWOP to retain more edges, MISIV to avoid degree zero vertices and MEC to produce large orthology sets. This divergence was not contrived; indeed, while trying to satisfy one objective criterion, as a secondary policy we tried to satisfy the other criteria whenever there was a choice, e.g., step 3 of the MISIV algorithm.

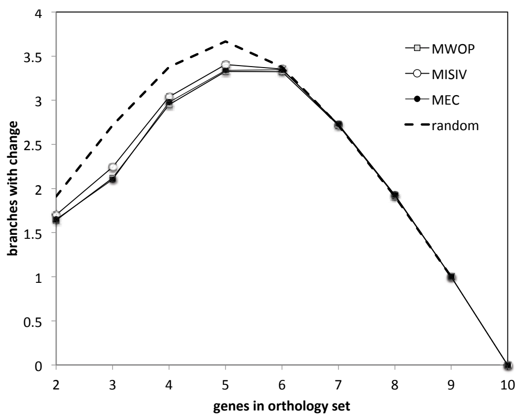


Fig. 4. Compatibility of orthology sets with phylogeny

3 Application to 10 Plant Genomes

We used data drawn from 10 core eudicotyledon genomes available in CoGe. While all genomes have been publicly available for at least one year, some lack a primary publication. To avoid infringing on the release conditions claimed by some of the sequencing groups, we will not identify the genomes we sampled. For the same reason, although we will make use of the phylogenetic relationships among these plants, which respects the current consensus [9], we will not present the phylogeny explicitly. However, for a list of sequenced eudicotyledon genomes in CoGe, consult `http://genomevolution.org/r/3119`; for a list of sequenced plant genomes, `http://genomevolution.org/r/3118`.

We used SynMap to produce sets of synteny blocks between all 45 pairs of genomes, and additionally within each of the five descendants of WGD events. Four different data sets were created using the QuotaAlign option [10], by varying the minimum block length parameter through the four values 1,2,3 and 5. We used the default options for all other settings. An average of more than 10,000 pairs of homologous genes were inferred in the runs for minimum block size 5, for example, involving slightly less than 10,000 distinct genes in each genome.

We extracted all the homology relations from all the synteny blocks in each pair, and put them all together, along with the paralogies within the five WGD descendants, to form the graph H. This had some 160,000 vertices and 485,000 edges, falling into 16,300 disjoint components.

3.1 Percolation, Tangles and Run Time

In order to pick up as many true orthologies as possible, SynMap will unavoidably have some low rate of spurious identification of homologs. This has little consequence for pairwise genomic correspondences for which SynMap was built, but when multiple sets of correspondences are merged to generate the set H, a kind of local percolation phenomenon manifests itself as a large tangle of genes, most of which are not closely related, but are contained in the same component. It is inherent in this non-zero rate of spurious homology, even though it is low, that the larger the tangle, the more likely it is to grow as the number of genomes increases. With our 10 genomes and for minimum block size 5, for example, 29,000 of the 485,000 edges, or 6 %, were involved in one large tangle, as shown in Table 1. With minimum block size 1, the tangle contained almost 90,000 edges.

Table 1. Distribution of component size, showing one large tangle

	frequencies of size of component in H					
edges in component	1	2-14	15-105	106-300	301-1000	29,134
frequency	2550	2776	9,350	523	26	1

Tuning SynMap to be more conservative might lead to smaller tangles, but the systematic experimentation that would be required is beyond the scope of the present study. Note that the ortholog sets we require contain at most 15 genes all linked by 105 edges, one gene from each genome or possibly two from each descendant of a WGD event. The current implementation of MEC requires excessive computing time when the number of edges exceeds a few hundred. To avoid this, we simply preprocess the large components in H, filtering out homologs with weights w below a threshold w^*. The value of $w*$ is raised until the tangle and other large components break up into pieces smaller than 300 edges. This step is unnecessary for MWOP and MISIV but for comparability we use the same reduced input graph for all the methods.

The worst case run time for the auction method we use in our MWOP implementation is $O(w'|V|^3)$, where w' is an integer representing the maximum weight on any edge in the current matching step - the original two-decimal input weights having been converted at the outset to integers. Then $O(Nw'|V|^3)$ is the run time for our entire algorithm, where N is the total number of genomes being matched. Since $|E| \leq 300$, the number of vertices $|V|$ is small and the algorithm runs in a few milliseconds.

Similarly, the small size of $|V|$ means that implementation of an efficient maximum weight matching, say $O(|V|^3)$, within the MISIV algorithm is unnecessary for the current comparison, even when the optional extra step 3 is executed. The algorithm generally runs in less than a second, but can occasionally take a few seconds.

The exhaustive greedy search at every step of the MEC algorithm is time consuming, with even the Warshall algorithm for finding the transitive closure requiring $O(|V|^3)$ time, or $O(|E|^2|V|^3)$ for finding all edges to remove, each time checking all $|E|$ of them. This works out to about 30 seconds per run.

Recall that each of the algorithms was applied to 15-20,000 homology graph components, for four different block length thresholds.

4 Results

Figure 2 shows that MWOP retains marginally more edges than the other two methods, but that MISIV creates far fewer degree-zero vertices. For minimum block length 5, the total number of deleted edges is about 8 % of all edges and total gene deletions about 3.5 %. The increasing number of edges as a function of minimum block length is an artifact of tangle size; in fact there may be marginally more edges with smaller minimum block lengths, but the tangle is much larger, so that when this is resolved by the method in Section 3.1, there are fewer edges input to our algorithms for smaller minimum block lengths.

Figure 3 shows that MEC packs the same number of edges into fewer, bigger (and therefore better for biological purposes) orthology sets.

To test to what extent the gene sets computed are compatible with the known phylogeny, we computed for each gene the minimum number of times it would have to be inserted and deleted on the tree (e.g. a gene set that includes a gene

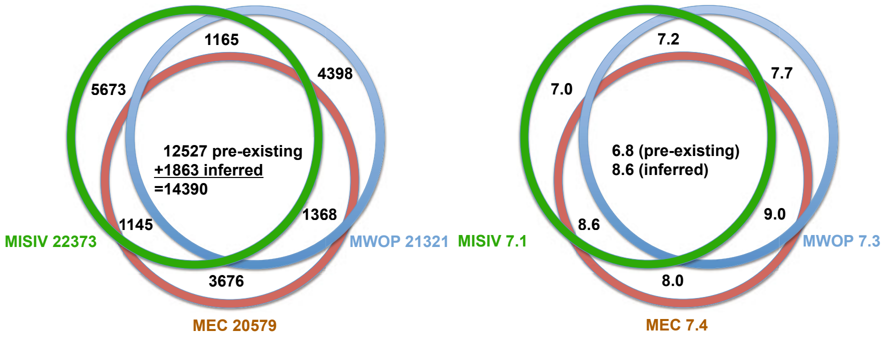


Fig. 5. Proportional Venn diagrams of the number (left) and mean size (right) of the orthology sets recovered by three methods

from all the genomes in the phylogeny requires zero insertions and deletions). The average of this value for each set size is shown in Figure 4. There is a strong phylogenetic signal appearing for small to moderate sized orthology sets, when compared to orthology sets with genes allocated at random to a given number of genomes, i.e., the inferred orthology sets implied a smaller number of insertions and deletion on the tree than random data.

Figure 5 depicts a proportional Venn diagram of the number of orthology sets shared among the results of the three methods. It can be seen that about 60% of the sets recovered by any of the methods were already orthogonal in the input data, and another 8 - 9% are also found by both of the other two methods. An additional 5 - 6.5 % of sets from one method are shared with only one of the other two. The MEC method can be seem as producing the smallest proportion of "idiosyncratic" sets, around 18% (or 45 % of the inferred, or non-pre-existing sets), and MISIV the most, more than 25% (or 57 % of the inferred, or non-pre-existing sets). MEC in this sense is somewhat of a compromise between the extreme "save homologies" goal of MWOP and the "save genes" aim of MISIV.

5 Conclusions

At least for the aims of gene-order alignment, the larger orthology sets produced by MEC lead to our recommendation of MEC as the method of choice. This must eventually be subjected, of course, to validation against curated orthology sets.

We can also recommend using minimum block length of at least 5 with SynMap, as Figures 2 and 3 show improved results according to several parameters with increasing block length. At some point, however, higher minimum block length will reduce the number of edges produced by SynMap.

At present, we consider only WGD events in the immediate lineage of single genomes in the data set; paralogs dating from ancient polyploidies shared by all the genomes present will already have been resolved in the output of SynMap. Future developments should allow several genomes to share a WGD event in their ancestry, and for multiple WGD events to occur in the lineage of a single genome.

The problem of tangles is a minor annoyance, quantitatively, in our work, but it could become much worse as increasing numbers of genomes are included in the analysis. Avenues available for attenuating this include tuning SynMap to be more conservative, increasing minimum block length, and more stringent criterion for WGD-origin paralogies in H.

The ultimate validation of the orthology sets produced by the different methods will involve the careful study of many of the sets, especially those that are not captured by all methods, and their comparison with biologically curated sets of homologs. This evaluation, as well as a comparison of the phylogenetic consequences of using different methods for OMG and the identification of genomic data most susceptible to tangles, is hindered by the severe interpretations of the Fort Lauderdale convention imposed by many genome projects. These constraints surpass those recognized by the broader community [13] and are not necessarily respected by journals [14].

References

1. Deniélou, Y.P., Sagot, M.-F., Boyer, F., Viari, A.: Bacterial syntenies: an exact approach with gene quorum. BMC Bioinformatics (in press, 2011)
2. Fostier, J., et al.: A greedy, graph-based algorithm for the alignment of multiple homologous gene lists. Bioinformatics 27, 749–756 (2011)
3. Shulaev, V., et al.: The genome of woodland strawberry (*Fragaria vesca*). Nat. Genetics 43, 109–116 (2011)
4. Zheng, C., Sankoff, D.: Gene order in Rosid phylogeny, inferred from pairwise syntenies among extant genomes. In: Chen, J., Wang, J., Zelikovsky, A. (eds.) ISBRA 2011. LNCS, vol. 6674, pp. 99–110. Springer, Heidelberg (2011)
5. Lyons, E., Freeling, M.: How to usefully compare homologous plant genes and chromosomes as DNA sequences. Plant J 53, 661–673 (2008)
6. Lyons, E., et al.: Finding and comparing syntenic regions among Arabidopsis and the outgroups papaya, poplar and grape: CoGe with rosids. Plant Phys. 148, 1772–1781 (2008)
7. He, G., Liu, J., Zhao, C.: Approximation algorithms for some graph partitioning problems. Journal of Graph Algorithms and Applications 4, 1–11 (2000)
8. Warshall, S.: A theorem on boolean matrices. Journal of the ACM 9, 11–12 (1962)
9. Angiosperm Phylogeny Group.: An update of the Angiosperm Phylogeny Group classification for the orders and families of flowering plants: APG III. Botanical Journal of the Linnean Society 161, 105–121 (2009)
10. Tang, H., et al.: Screening synteny blocks in pairwise genome comparisons through integer programming. BMC Bioinformatics 12, 102 (2011)
11. Mestre, J.: Maximum weight matching via auctions (2009), http://www.mpi-inf.mpg.de/departments/d1/teaching/ws09_10/Opt2/handouts/lecture1.pdf
12. Nisan, N.: Auction algorithm for bipartite matching. *Algorithmic Game Theory/Economics* (2009), http://agtb.wordpress.com/2009/07/13/auction-algorithm-for-bipartite-matching/
13. Birney, E., et al.: Prepublication data sharing. Nature 461, 168–170 (2009), Ehrlich SD
14. Nature Editors: Sacrifice for the greater good? Nature 421, 875 (2003)

Author Index